VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2017 **Marek Jeriga**

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Programování sekvenčních úloh u programovatelných automatů

Sequential Tasks Programming in Control Systems with Programmable Controllers

2017 **Marek Jeriga**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Marek Jeriga**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2612R041 Řídicí a informační systémy

Téma: Programování sekvenčních úloh u programovatelných automatů
Sequential Tasks Programming in Control Systems with Programmable Controllers

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Rozbor možností automatizované tvorby řídicí aplikace ve vybraném nástroji pro programování programovatelných automatů.
2. Rozbor možností programování sekvenčních logických úloh. Funkční popis sekvenčních úloh.
3. Návrh zjednodušeného způsobu popisu sekvenční řídicí úlohy.
4. Návrh a realizace řídicí aplikace vybrané sekvenční úlohy.
5. Ověřené funkčnosti a efektivity návrhu.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] BERGER, Hans. *Automating with SIMATIC.* 5th edition. Erlangen, Germany: Publicis Publishing, 2013, 284 p. ISBN 978-3895783876.
[2] BERGER, Hans. *Automating with SIMATIC S7-1500: Configuring, Programming and Testing with STEP 7 Professional.* Hardcover, 2014. ISBN-13: 978-3895784040.
[3] Technická dokumentace k systému Simatic.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *28. dubna 2017*

.....
podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Jiřímu Koziorkovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tato bakalářská práce rozebírá možnosti automatizované tvorby řídicí aplikace pro programovatelné logické automaty ve zvoleném vývojovém prostředí TIA Portal V13, export a import jednotlivých částí programu, a také programování sekvenčních logických úloh, sekvenční logické obvody používané pro programování programovatelných logických automatů a systematické postupy vhodné pro programování sekvenčních úloh. V praktické části je vytvořen nástroj pro zjednodušený způsob popisu sekvenční úlohy pomocí programu Microsoft Excel 2016. Nástroj je otestován pomocí úlohy, která je navržena vytvořeným nástrojem v prostředí Microsoft Excel 2016, zrealizována a posléze naimportována do vývojového prostředí TIA Portal V13. V prostředí TIA Portal V13 je úloha otestována jak pomocí simulačního softwaru S7-PLCSIM, tak také pomocí reálného PLC.

Klíčová slova

PLC, TIA Portal, SIMATIC, Excel, makra, STL, sekvenční, SFC

Abstract

This bachelor thesis discusses the possibility of automated creation of control applications for programmable logic controllers in the selected development application TIA Portal V13, export and import of individual parts of the program, programming sequential logic tasks, sequential logic circuits used for programming of programmable logic controllers and systematic procedures appropriate for sequential programming tasks. In the practical part, a tool for a simplified way of describing a sequential task is created by using Microsoft Excel 2016. The tool is tested by using a task that is designed by a created tool in Microsoft Excel 2016, implemented and then imported into the TIA Portal V13 development environment. In TIA Portal V13, the task is tested with the S7-PLCSIM simulation software and real PLC.

Key words

PLC, TIA Portal, SIMATIC, Excel, macros, STL, sequential, SFC

Obsah

Seznam použitých zkratek

Zkratka	Význam
PLC	Programmable Logic Controller
TIA Portal	Totally Integrated Automation Portal
SFC	Sequential Function Chart
IEC	International Electrotechnical Commission
HMI	Human Machine Interface
SCADA	Supervisory Control And Data Acquisition
SCL	Structured Control Language
LAD	Ladder diagram
FBD	Function Block Diagram
STL	Statement List
API	Application Programing Interface
MS Excel	Miscrosof Excel
HW	Hardware

Seznam ilustrací a seznam tabulek

Úvod

Bakalářskou práci na téma programování sekvenčních úloh u programovatelných automatů jsem zvolil především z toho důvodu, že práce s programovatelnými automaty mě baví a v budoucnu bych se jim chtěl dále věnovat. Hlavním cílem práce bylo najít vhodné způsoby popisu sekvenčních úloh, zjistit, jak lze do zvoleného programovacího prostředí importovat, respektive z něj exportovat jednotlivé části řídicí aplikace, vytvořit zjednodušený popis sekvenční řídicí úlohy a celkově tak zjednodušit tvorbu celé řídicí aplikace.

V první kapitole jsou rozebrány možnosti automatizované tvorby řídicí aplikace v prostředí pro tvorbu řídicí aplikace programovatelných logických automatů TIA Portal V13. Je zde stručně popsáno vývojové prostředí TIA Portal V13, nástroj STEP 7 Professional V13, rozhraní TIA Portal Openness a také Microsoft Excel. Dále je zde uvedeno, které části lze z prostředí TIA Portal V13 exportovat a zpětně importovat.

Druhá kapitola se zabývá samotným programováním sekvenčních úloh, jsou zde rozebrány sekvenční logické funkce a detailněji některé sekvenční logické obvody, především ty, které se běžně používají pro programování sekvenčních úloh u programovatelných logických automatů. Dále uvádí do problematiky návrhu sekvenčního programu a ukazuje některé z možností, jak lze systematicky postupovat při návrhu úloh tohoto typu. Nejvíce části je věnováno programovacímu jazyku SFC, který je podporovaný normou IEC 61131-3 a některé vývojové prostředí dokonce umožňují tento jazyk, nebo jeho obdobu, využívat při tvorbě programu. Stručně jsou ukázány i další postupy, jako P/T Petriho sítě a také Mealyho a Mooreův automat.

Následující kapitola ukazuje tvorbu nástroje pro zjednodušení tvorby řídicí aplikace, dává návod k použití tohoto nástroje a také k následnému nahrání souborů vytvořených v prostředí Microsoft Excel 2016 do prostředí TIA Portal V13 a také opačný postup.

Čtvrtá kapitola se zabývá samotným návrhem a realizací řídicí aplikace. Je zde k nalezení zadání zvolené sekvenční úlohy, její grafické znázornění, návrh pomocí sekvenčního diagramu SFC a především samotná realizace úlohy a následné otestování jak pomocí simulačního softwaru S7 - PLCSIM, tak také pomocí reálného PLC řady S7-1500.

1 Možnosti automatizované tvorby řídicí aplikace v prostředí TIA Portal V13

1.1 TIA Portal V13

Jako nástroj pro programování programovatelných automatů byl zvolen, vzhledem k tomu, že toto prostředí je využíváno i pro výuku na naší fakultě, systém TIA Portal V13 společnosti Siemens, s.r.o. umožňující nejen programovat PLC, ale také navrhovat operátorské panely HMI a počítačové vizualizační systémy SCADA, k čemuž bylo v minulosti nezbytné mít další programy.

1.1.1 STEP 7 Professional V13

STEP 7 je nástroj v prostředí TIA Portal, který umožňuje programovat řídicí systémy SIMATIC. STEP 7 V13 je nabízen ve dvou verzích, a to STEP 7 Basic a STEP 7 Profesional, hlavním rozdílem mezi těmito dvěma verzemi je to, že verze Profesional umožňuje programovat všechny PLC rodiny SIMATIC, zatímco verze Basic je určena primárně pro programování PLC řady S7-1200. Dalším rozdílem mezi verzemi Basic a Profesional je to, že verze Profesional podporuje všechny programovací jazyky dle normy IEC-61131-3 (strukturovaný text - SCL, liniové schéma - LAD, diagram funkčních bloků - FBD, seznam instrukcí - STL, GRAPH - sekvenční funkční graf), verze Basic podporuje jen některé jazyky (LAD, FBD a SCL). [1][2]

1.2 TIA Portal Openness

TIA Portal Openness je rozhraní, které je dodáváné firmou Siemens společně se softwarem SIMATIC STEP 7 V13 SP1, WinCC V13 SP1 a také k jejich novějším verzím, případně je možné toto rozhraní získat zdarma po registraci na webových stránkách firmy Siemens, nebo na vyžádání od technické podpory firmy Siemens. Díky tomuto doplňku je možné TIA Portal propojit s externí aplikací, která se může díky programovatelnému rozhraní API naprogramovat přesně podle konkrétních požadavků dané firmy. Siemens zdarma poskytuje ukázkové aplikace vyvinuté v Microsoft Visual Studio, které dávají návod, jak lze aplikaci naprogramovat. Openness umožňuje uživatelům, mimo jiné, automatické generování některých částí projektu a tím nejen výrazně usnadňuje a zrychluje práci programátora, ale také minimalizuje chyby při vytváření projektu. Rozhraní funguje tak, že přistupuje ke knihovnám s příponou .dll, díky kterým je umožněno ovládat funkce programu TIA Portal.[3][4]

Výčet některých funkcí, které umožňuje TIA Portal Openness:

- Zapnutí / vypnutí programu TIA Portal
- Otevření, zavření, uložení projektu
- Kompilace změn hardwaru i softwaru
- Připojení / odpojení PLC
- Export / import bloků
- Generování bloků ze zdrojového souboru
- Generování zdrojového souboru z bloků (pro jazyky STL a SCL)
- Export / import proměnných ve formátu XML [4]

1.3 Microsoft Excel

Microsoft Excel (dále jen Excel) je tabulkový procesor vyvinutý firmou Microsoft. Excel je součástí kancelářského balíku Microsoft Office. O tomto tabulkovém procesoru se v mé bakalářské práci zmiňuji hlavně proto, že tento software byl zvolen jako nedílná součást při automatizované tvorbě řídicí aplikace. Excel umožňuje číst a případně pozměnit ty části projektu, které lze z prostředí TIA Portal exportovat a následně zpětně importovat. V následujících kapitolách bude ukázáno, jaké části projektu lze z prostředí TIA Portal exportovat a jak poté do tohoto prostředí můžeme nahrát upravený či původní soubor.

1.3.1 Export souborů z TIA Portalu do Excelu

TIA Portal V13 umožňuje export programových bloků (organizační blok, funkční blok, funkce, datový blok) a také export tabulky proměnných. U programových bloků export závisí na tom, v jakém programovacím jazyce je program napsán, protože export je možný pouze pro textové programovací jazyky STL a SCL, pro grafické programovací jazyky export programu možný není. PLC řady SIMATIC S7-300 a S7-400 umožňují změnu programovacího jazyka z grafických LAD a FBD na textový programovací jazyk STL. Při využití tohoto postupu při exportu však musí být brán ohled na to, že ne všechny funkce, které lze použít pro grafické programovací jazyky, lze následně převést na textový programovací jazyk STL. Převod ať už grafických programovacích jazyků LAD, FBD a GRAPH, či textového programovacího jazyku STL na textový programovací jazyk vyšší úrovně SCL není v prostředí TIA Portal možný. PLC řady SIMATIC S7-1200 a S7-1500 změnu programovacího jazyku z grafického na textový, ani naopak, neumožňují.

1.3.2 Import z Excelu do TIA Portalu

Tak jako je možné exportovat programové bloky a proměnné z TIA Portalu do Excelu, je samozřejmě možný i opačný postup, což může značně usnadnit celý postup tvorby řídicí aplikace. Do TIA Portalu V13 je možné vkládat programové bloky napsané v textových programovacích jazycích STL a SCL. Pokud je vložen blok napsaný v jazyce STL a pracuje se s PLC řady SIMATIC S7-300 nebo S7-400, je možné poté programovací jazyk změnit na grafický LAD nebo FBD. Důležité pro vkládání souborů do TIA Portalu je dodržet správný formát zápisu proměnných i programových bloků.

2 Programování sekvenčních logických úloh

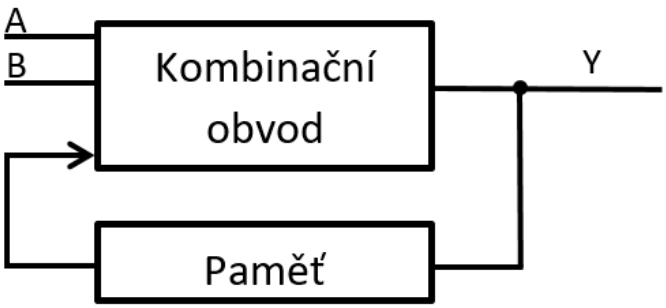
2.1 Sekvenční logické funkce

Sekvenční logické funkce jsou funkce, u kterých jejich výstupní hodnota závisí jak na hodnotách vstupních proměnných, tak také na aktuálním stavu systému a tím se odlišují od kombinačních funkcí, u kterých je výstupní hodnota pevně dána pouze hodnotami, které jsou přivedeny na jejich vstup. Aktuální stav sekvenční logické funkce může být dán pořadím vstupních hodnot, ale také například počtem opakování či časem. O sekvenční funkci lze hovořit vždy, když jsou zpracovávány vstupní hodnoty i stav obvodu. [4]

Sekvenční funkce se v praxi používají v hojné míře, a to například v různých typech výrobních podniků, kde výroba probíhá za pomoci výrobních robotů, nebo při řízení budov. Při programování by se do programu měli sekvenční logické funkce zanášet vždy vědomě, jinak se ve výsledku může stát, že budou způsobovat více škody než užitku a budou způsobovat chyby. [4]

2.2 Sekvenční logické obvody

Sekvenční logické obvody reprezentují sekvenční logické funkce a jsou složeny z kombinačního obvodu a paměťového prvku. Výstupní hodnota je závislá na vstupních proměnných a na předchozím stavu obvodu. Sekvenční logické obvody lze rozdělit na synchronní a asynchronní. U synchronních logických obvodů je navíc hodinový vstup, kterým se řídí změna výstupních hodnot, které se mění až s příchodem hodinového signálu, a ne ihned po změně vstupních hodnot. Asynchronní logické obvody hodinový vstup nemají a výstupní hodnoty se mění okamžitě se změnou vstupních hodnot. Mezi sekvenční logické obvody řadíme například klopné obvody, čítače, časovače a posuvné registry.



Obrázek 2.1: *Blokové schéma sekvenčního obvodu*

2.2.1 RS klopný obvod

Klopný obvod RS patří mezi nejzákladnější klopné obvody, jeho funkce je znázorněna v tabulce 2.1. Obvod má vstupy R a S, které určují, jaká hodnota bude na výstupu. Jsou-li oba vstupy v logické 0,

hodnota na výstupu Q se nemění. Pokud je na vstupu R logická 1 a na výstupu Q je také logická 1, výstup Q se překlopí do stavu logická 0, pokud výstup Q již byl v hodnotě logická 0, jeho hodnota se nezmění. Obdobně funguje i vstup S, který ale naopak, pokud je v logické 1 a výstup Q v logické 0, překlopí výstup Q do logické 1, v případě že výstup Q již byl v logické 1, tak se nezmění. Pokud na oba vstupy, tedy na R i S je přivedena logická 1, je jeho výstup dán tím, jak je obvod vnitřně realizován. [6]

O obvodu RS se hovoří jako o asynchronním klopném obvodu, jeho výstupní hodnota se tedy mění okamžitě po změně vstupních hodnot a není závislá na čase. Pokud obvod doplníme o hodinový signál, lze hovořit o klopném obvodu RST, který je synchronní a výstup se mění pouze tehdy, když je na vstupu T přivedena logická 1.

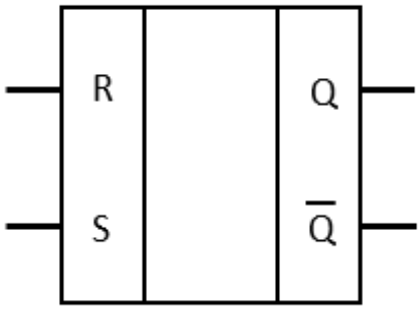
Tabulka 2.1: Pravdivostní tabulka klasického RS klopného obvodu z hradel NOR

R	S	Q_{t+1}	$\bar{Q}_{t+1}$
0	0	Q_t	$\bar{Q}_t$
0	1	1	0
1	0	0	1
1	1	X	X

V programu TIA Portal je klopný obvod RS s prioritou pro vstup S, to znamená, že pokud jsou obě vstupní hodnoty R i S v logické 1, je na výstupu Q vždy logická 1. Jinak je obvod funguje stejně jako klasický RS, tak jak je znázorněno v tabulce 2.2. [8]

Tabulka 2.2: Pravdivostní tabulka RS klopného obvodu dostupného v programu TIA Portal

R	S	Q_{t+1}	$\bar{Q}_{t+1}$
0	0	Q_t	$\bar{Q}_t$
0	1	1	0
1	0	0	1
1	1	1	0



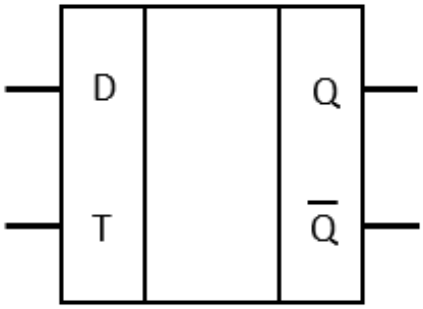
Obrázek 2.2*: Schématická značka RS klopného obvodu*

2.2.2 D klopný obvod

Klopný obvod D má vstup D, který udává hodnotu na výstupu Q. Je-li tedy vstup D v logické 1, je i výstup Q v logické 1. Obvod bývá realizován jako synchronní, tedy s hodinovým vstupem T. Obvod poté funguje tak, že změna na vstupu D se na výstupu Q projeví jen tehdy, pokud je na vstupu T logická 1. Obvod reaguje na vzestupnou hranu. [7]

Tabulka 2.3: *Pravdivostní tabulku D klopného obvodu*

D	Q_{t+1}	$\bar{Q}_{t+1}$
0	0	1
1	1	0



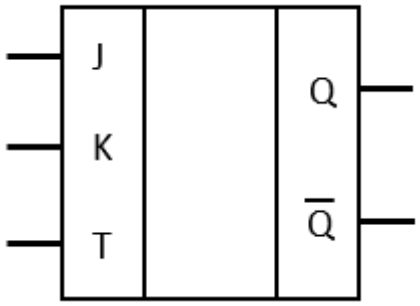
Obrázek 2.3: *Schématická značka D klopného obvodu s hodinovým vstupem*

2.2.3 JK klopný obvod

Klopný obvod JK je obdobou klopného obvodu RS a jasně udává, co se bude dít, pokud jsou oba vstupy v logické 1. Pokud jsou oba vstupy v logické 0 nebo v logické 1, výstupní hodnota zůstává nezměněna. Logická 1 přivedena na vstup J způsobuje změnu výstupu Q na logickou 1. Logická 1 přivedena na vstup K zase způsobuje změnu výstupu Q na logickou 0. Obvod se realizuje jako synchronní, tedy se vstupem pro hodinový signál T, který řídí, kdy má výstup Q reagovat na změnu hodnot vstupů J a K. [7]

Tabulka 2.4: *Pravdivostní tabulku JK klopného obvodu*

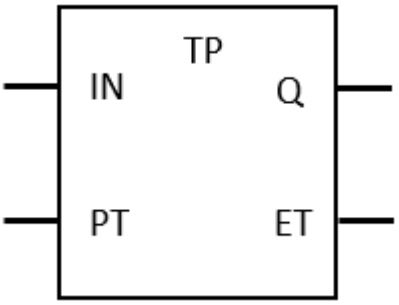
J	K	Q_{t+1}	$\bar{Q}_{t+1}$
0	0	Q_t	$\bar{Q}_t$
0	1	0	1
1	0	1	0
1	1	$\bar{Q}_t$	Q_t



Obrázek 2.4: *Schématická značka JK klopného obvodu s hodinovým vstupem*

2.2.4 **Časovač TP**

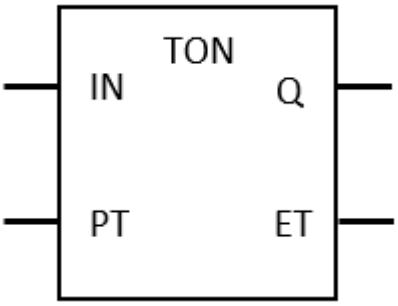
Časovač TP je základní časovač, který má, stejně jako časovače TON a TOF, vstupní proměnné IN a PT a výstupní proměnné Q a ET. Vstup IN řídí činnost časovače, vstupem PT se nastaví délka pulzu, který je na výstupu Q. Na výstupu ET lze sledovat aktuální stav časovače vyjádřený časem. Časovač TP funguje tak, že po tom, co je na vstup IN přivedena logická 1, je zároveň s její náběžnou hranou generován pulz na výstupu Q o délce, která je předem nastavena na vstupu PT. Impulz vždy proběhne celý a nejde jej přerušit a to bez ohledu na to, jakou hodnotu má vstup IN. [8][12]



Obrázek 2.5: *Schématická značka časovače TP*

2.2.5 **Časovač TON**

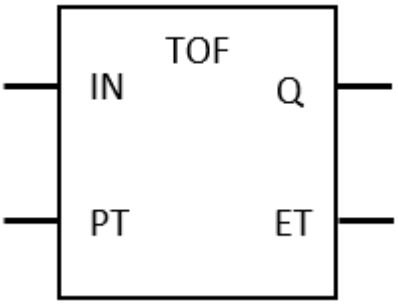
Časovač TON je časovač se zpožděním náběžné hrany. Po přivedení logické 1 na vstup IN se výstup Q překlopí do logické 1 až poté, co uplyne čas, který je nastaven na vstupu PT. Zpátky do logické 0 se vrátí výstup Q až tehdy, když na vstupu IN bude nulová hodnota. Časovač bude resetován po přivedení logické 0 na vstup IN a to i v případě, že se zatím nestihl výstup Q překlopit do logické 1. Časovač začne znovu čítat po přivedení logické 1 na vstup IN. Výstup ET funguje stejně jako u časovače TP a ukazuje aktuální hodnotu časovače. [8][12]



Obrázek 2.6: *Schématická značka časovače TON*

2.2.6 **Časovač TOF**

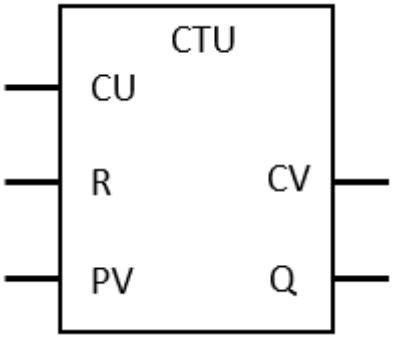
Časovač TOF je časovač se zpožděním sestupné hrany. Pokud je na vstup IN přivedena logická 1, výstup Q se překlopí do logické 1. Po přivedení logické 0 na vstup IN začne časovač čítat do hodnoty, která je nastavená na PT, poté se Q překlopí zpět do logické 0. Při opětovném přivedení logické 1 na vstup IN je časovač resetován a dále vše proběhne tak, jako v předchozím případě. Na výstupu ET lze sledovat aktuální hodnotu časovače. [8][12]



Obrázek 2.7: *Schématická značka časovače TOF*

2.2.7 **Čítač CTU**

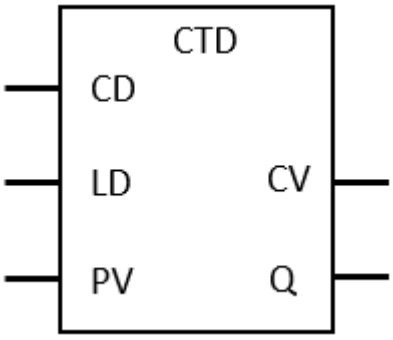
Čítač CTU je čítač nahoru a má vstupy CU pro čítání nahoru, R pro resetování, PV pro nastavení hodnoty do které se má čítat a výstupy CV pro zobrazení aktuální hodnoty čítače a výstup Q, který podává informaci o tom, zda čítač již načítal do požadované hodnoty. S každým pulzem, který dorazí na vstup CU se hodnota čítače zvyšuje, v okamžiku, kdy čítač dočítá do hodnoty, která je nastavena na PV, tak se výstup Q překlopí do logické 1. V případě, že je na vstup R přivedena logická 1, je čítač resetován. Čítání může začít znovu až tehdy, když je na vstupu R logická 0. [8][13]



Obrázek 2.8: *Schématická značka čítače CTU*

2.2.8 **Čítač CTD**

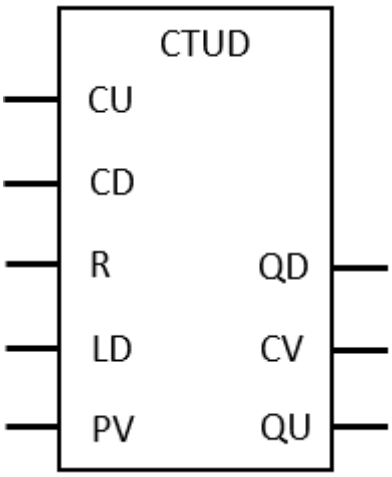
Čítač CTD je obdobou čítače CTU s tím rozdílem, že čítá dolů. Jeho vstupy jsou CD pro čítání dolů, LD, který je obdobou vstupu R u čítače CTU a PV, kterým se nastavuje hodnota, od které čítač bude čítat dolů. Výstupy jsou CV pro zobrazení aktuální hodnoty a Q, kde se zobrazí logická 1 po načítání do nuly. Čítač má takovou hodnotu, jakou nastavíme na vstup PV, příchodem pulzu na vstup CD se hodnota čítače zmenší o 1. Na výstupu CV se zobrazuje aktuální hodnota. Pokud je hodnota čítače nulová, na výstupu Q je logická 1. Pokud je na LD přivedena logická 1, časovač a tedy i hodnota výstupu CV je resetována na hodnotu nastavenou na vstupu PV. Čítač může znovu začít čítat až tehdy, je-li na vstupu LD logická 0. [8][13]



Obrázek 2.9: *Schématická značka čítače CTD*

2.2.9 **Čítač CTUD**

Čítač CTUD je obousměrný čítač. Vzhledem k tomu, že kombinuje předchozí dva čítače, tak i jeho vstupy a výstupy jsou kombinací předchozích čítačů, tedy čítačů CTU a CTD. Vstupy jsou CU, CD, R, LD a PV a plní stejnou funkci jako u čítač CTU, respektive CTD. Výstupy čítače CTUD jsou QU, který je v logické 1 pokud čítač načítal zvolenou hodnotu, QD, který je v logické 1 v případě, že čítač má nulovou hodnotu a CV, na kterém lze sledovat aktuální hodnota čítače. Pokud je přiveden pulz na vstup CU, hodnota čítače se inkrementuje, pokud je přiveden pulz na CD, provede se naopak dekrementace čítače. Pokud je na vstup R přivedena logická 1, čítač je nastaven na nulovou hodnotu. Dokud není na vstupu R znovu logická 0, čítač nemůže čítat a změny hodnot na CU, CD a LD jsou ignorovány. Přivedením logické 1 na vstup LD je hodnota čítače CV nastavena na hodnotu vstupu PV. Dokud je LD v logické 1, čítač nemůže čítat a změny hodnot na CU a CD nemají vliv na čítání. [8][13]



Obrázek 2.10: *Schématická značka čítače CTUD*

2.3 Návrh sekvenčního programu

Jednoduché sekvenční úlohy lze řešit samozřejmě intuitivně, avšak u složitějších úloh, na kterých se pracuje delší dobu nebo se na nich pracuje ve skupině, je vhodné použít systematický postup a využít jednu z metod, které jsou vhodné pro programování sekvenčních úloh.

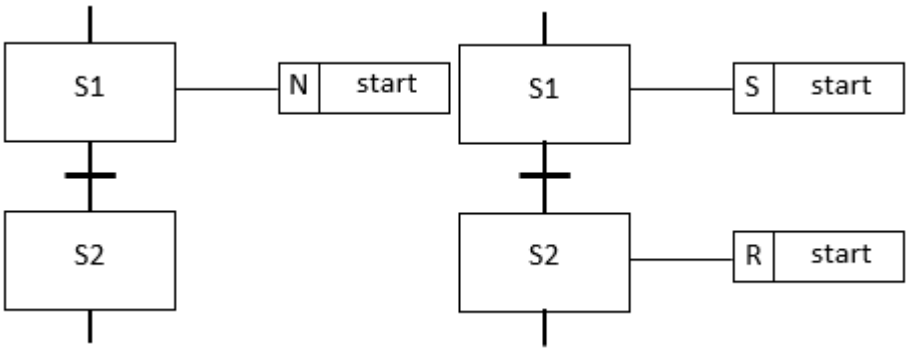
Společné pro všechny níže popsané možnosti sekvenčního programování je to, že je úloha rozdělena na několik oddělených událostí, které jsou prováděny postupně. V oddělených událostech lze program popsat akcemi, což jsou příkazy nebo části programu ovlivňující stav proměnných. [14]

2.3.1 Sequential Function Chart - SFC

SFC je programovací jazyk podporovaný normou IEC 61131-3, vhodný zejména pro návrh sekvenčních úloh. Vzhledem k tomu, že je tato forma návrhu sekvenčních úloh oficiálně zanesena v normě IEC, bude v mé bakalářské práci rozebrána detailněji než ostatní možnosti. Tento programovací jazyk rozebírá program na menší části a popisuje, jak mezi těmito menšími částmi programu proudí data. Jazyk je definován jak graficky, tak textově, používán je však především v jeho grafické formě. Předchůdcem SFC je Grafcet, se kterým bývá SFC často zaměňován, Grafcet však danou úlohu popisuje méně detailně než SFC. Grafcet je s SFC velmi podobný, platí pro něj podobná pravidla a výsledný graf se skládá ze stejných částí jako u SFC. O popisu pomocí SFC či Grafcetu se často hovoří jako o přechodovém diagramu (grafu), nebo jako o stavovém diagramu. SFC tvoří kroky, což jsou jednotlivé stavy, které jsou zaznačeny symbolem obdélníku, ve kterých je program po určitou dobu, a přechody, které tvoří podmínku, aby program přešel z aktuálního stavu do stavu následujícího. Dvojitý obdélník představuje počáteční stav programu. Poslední obdélník v programu je jediný, za kterým nemusí být přechod. Do obdélníků se vepisuje název daného stavu, většinou číslem. Přechody jsou logické podmínky a v SFC jsou zaznačeny horizontální čarou, vedle které se uvádí podmínka pro přechod do dalšího stavu. Vodorovná čára mezi dvěma stavy se nazývá hrana. V programu se musí kroky a přechody střídat, nesmí za sebou být dva kroky ani dva přechody bez přerušení. [9][14][15][16]

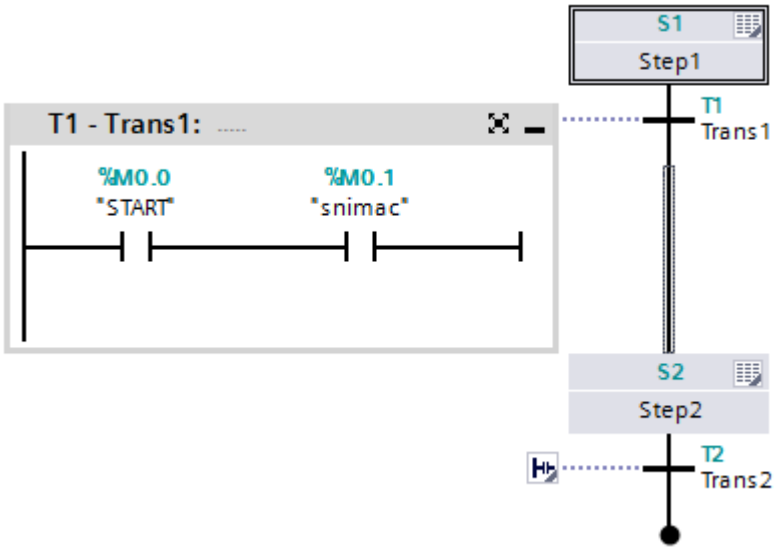
Kroky představují události a jsou znázorněny obdélníky, dvojitý obdélník představuje počáteční stav programu. Uvnitř obdélníků je naspán název stavu, většinou číslem. Každý krok má vlastní příznak kroku (step flag), což je proměnná nabývající hodnot 1 nebo 0 a to podle toho, jestli je krok právě aktivní (pokud je aktivní, má hodnotu logická 1). Příznak nese název konkrétního kroku, za který je připsána

tečka a písmeno X, příznak kroku S1 tedy bude vypadat takto: S1.X. Příznak kroku může být v programu použit jako proměnná nebo jako podmínka přechodu do dalšího kroku. Příznak doby, kdy je krok aktivní, je také přiřazen každému kroku. Označuje dobu, po jakou je daný krok aktivní a nese název konkrétního kroku, za kterým je připsána tečka a písmeno T, příznak doby, kdy je krok aktivní pro krok S1 bude vypadat následovně: S1.T. Při spuštění programu a také při aktivaci daného kroku je tento čas vynulován a po proběhnutí kroku je tento čas uložen. Stejně jako předchozí příznak kroku, tak i tento příznak doby setrvání v daném kroku může být použit v programu jako proměnná či jako podmínka pro přechod do dalšího kroku. Kroky mají buď přiřazenou nějakou akci, kterou vykonávají, nebo nemusí provádět žádnou akci a mohou jen čekat než je splněna podmínka pro přechod do dalšího kroku. Na obrázku 2.11 je ukázáno, jak například lze zapisovat a měnit hodnoty proměnných. Oba způsoby budou fungovat obdobně a to tak, že v kroku S1 bude mít proměnná start hodnotu logická 1, a v kroku S2 bude mít hodnotu logická 0. Kvalifikátor N totiž zapisuje do proměnné hodnotu příznaku aktuálního kroku, pokud je tedy krok aktivní, do proměnné se zapíše logická 1 a po přechodu do dalšího kroku bude proměnná kopírovat hodnotu příznaku předchozího kroku a také se změní na logickou 0. Kvalifikátory S a R fungují jako SET a RESET a proměnná tedy zůstane v dané hodnotě, dokud se jiným příkazem nezmění. V jednom kroku můžeme měnit i hodnoty více proměnných. Kroku lze přiřadit akci i pomocí jazyku strukturovaného textu, jakou možnost zvolíme, závisí na programátorovi. [14]



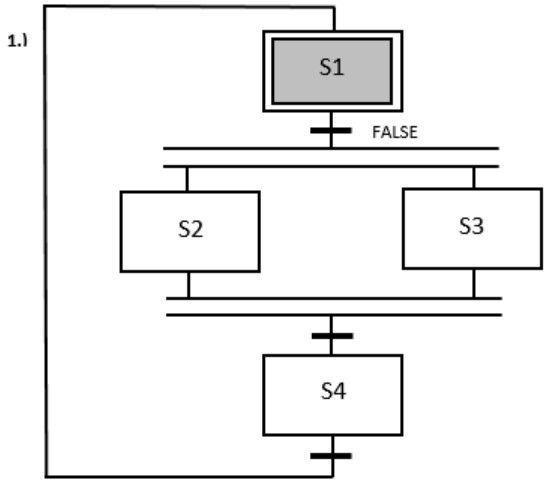
Obrázek 2.11: *Dva způsoby, jak realizovat stejnou akci pomocí SFC [14]*

Přechody jsou podmínky pro to, aby mohl program přejít do jiného kroku nebo několika jiných kroků a rozhoduje tedy o tom, který krok je aktivní. Pokud je podmínka přechodu splněna, předchozí krok se stane neaktivní a aktivním se stane krok následující. Podmínka přechodu může být napsána v jazycích LD, SCL, FBD, ale také pomocí konektoru směřujícím k přechodu. [14]

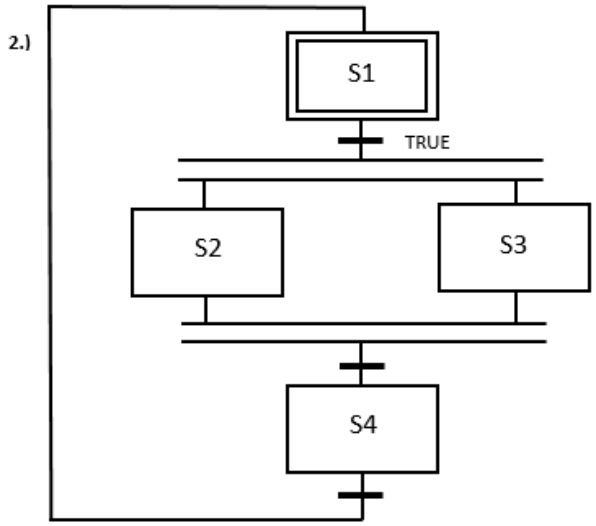


Obrázek 2.12: *Realizace přechodu v TIA Portal V13 v jazyku GRAPH vycházejícího z SFC*

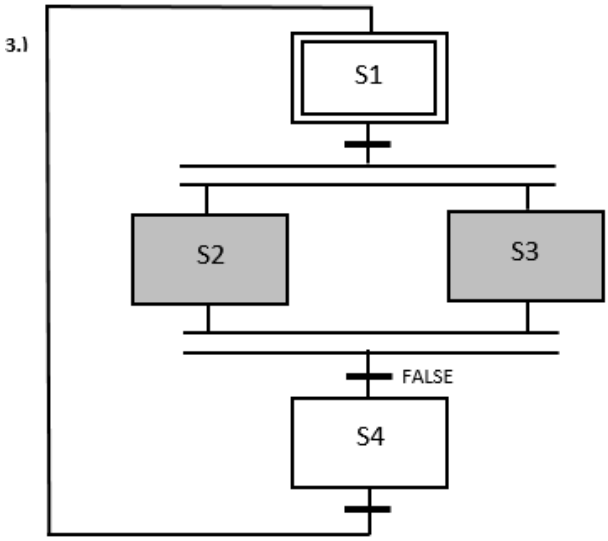
Na následujících šesti obrázcích je na jednoduchém příkladu vysvětleno, jak funguje jazyk SFC. Aktivní krok je vždy vybarven šedou barvou. Krok S1 je neaktivní od doby, kdy je přechod, který následuje za tímto krokem vyhodnocen jako TRUE (pravda), poté se stane aktivní krok S2 a krok S3. Tyto kroky jsou aktivní po dobu, než je následující přechod FALSE (nepravda), pokud bude přechod TRUE, aktivuje se krok S4. Po kroku S4 se program může vrátit zpět na počáteční krok S1 pokud přechod za krokem S4 vyhodnocen jako TRUE.



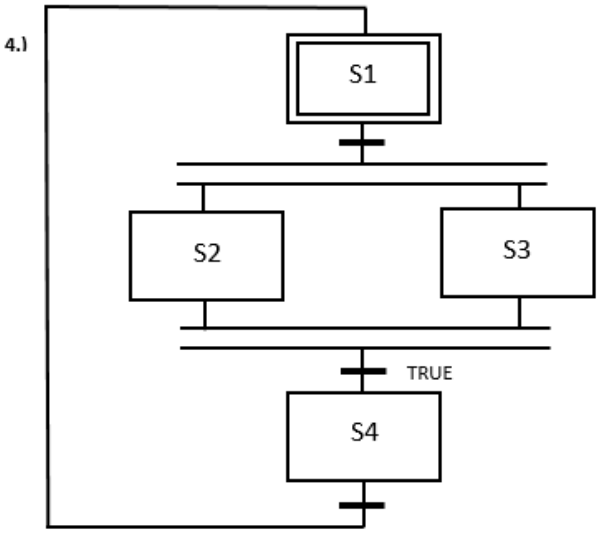
Obrázek 2.13: *Aktivní krok S1, přechod ve stavu FALSE [9]*



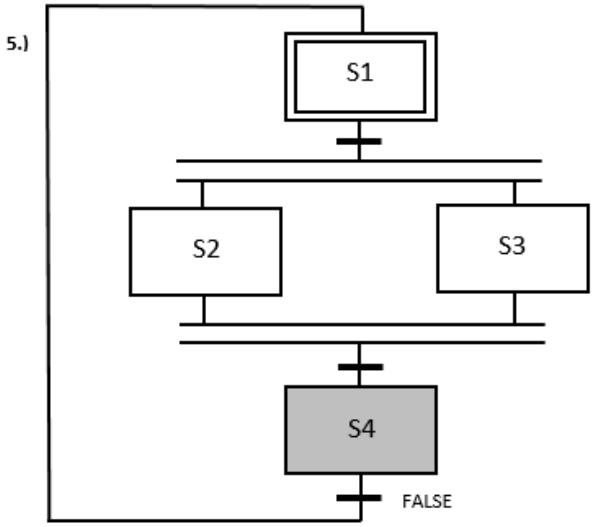
Obrázek 2.14: *Změna přechodu ze stavu FALSE na stav TRUE [9]*



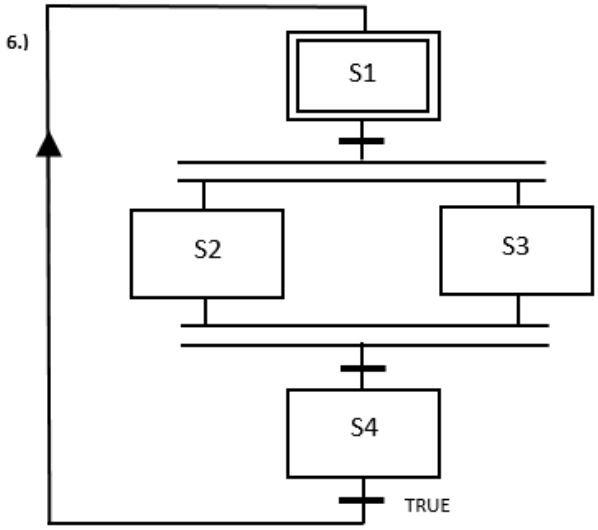
Obrázek 2.15: *Aktivní kroky S2 a S3, přechod ve stavu FALSE [9]*



Obrázek 2.16: *Změna přechodu ze stavu FALSE do stavu TRUE [9]*



Obrázek 2.17: *Aktivní krok S4, přechod ve stavu FALSE [9]*

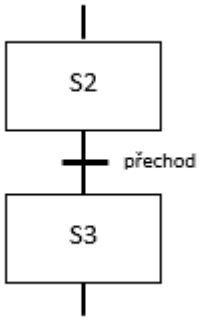


Obrázek 2.18: *Změna přechod ze stavu FALSE do stavu TRUE [9]*

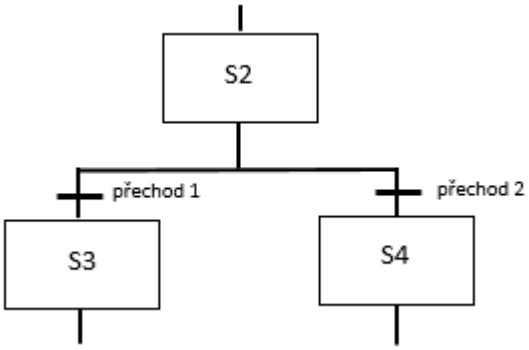
2.3.1.1 ***Možnosti posloupnosti kroků a přechodů***

Na následujících obrázcích bude ukázáno a stručně popsáno, jaké sekvence je možno tvořit v grafickém jazyce SFC. Na obrázku 2.19 je ukázána jednoduchá sekvence, kdy se kroky a přechody střídají v sérii za sebou, krok S3 bude aktivní, až bude přechod vyhodnocen jako TRUE (pravda). U některých úloh je však nutné, aby se graf rozděloval do více větví. Na obrázku 2.20 je ukázáno větvení, u kterého je možné, aby byl aktivní jen jeden krok. Vyhodnocování přechodů zde probíhá s prioritou zleva doprava, nejdříve je tedy vyhodnocen přechod 1, poté přechod 2. Pokud toto vyhodnocování není pro danou úlohu ideální, může programátor nastavit ručně prioritu vyhodnocování přechodů a to tak, jak je ukázáno na obrázku 2.21, nejnižší číslo má nejvyšší prioritu a je tedy vyhodnocováno jako první. Tak, jako můžeme graf rozdělit do více větví, lze ho samozřejmě i sjednotit, jak jde vidět na obrázku 2.22. Pokud zde bude aktivní jeden z kroků S3 nebo S4 a podmínka pro přechod bude pravdivá, aktivuje se krok S5. Je také možné, aby byly aktivní dva nebo více kroků

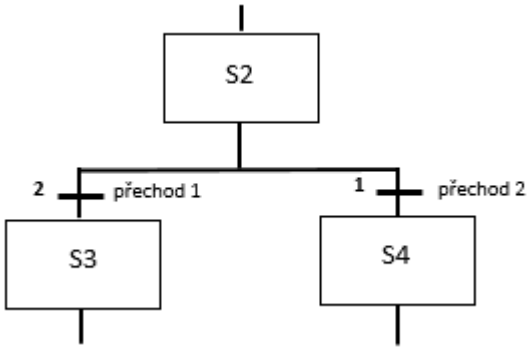
nezávisle na sobě, toho lze docílit pomocí takzvaného simultánního větvení, které je zobrazeno na obrázku 2.23. Simultánní větvení lze na první pohled poznat tak, že tam kde toto větvení začíná, je dvojitá vodorovná čára. Toto větvení funguje tak, že pokud je splněna podmínka pro přechod, aktivují se kroky S3 a S4. Obdobně funguje i sjednocení simultánního větvení, které je na obrázku 2.24. Kroky S3 a S4 musejí být aktivní, podmínka pro přechod musí být splněna a poté bude aktivní krok S5. Na obrázku 2.25 je zobrazena sekvenční smyčka, pro lepší přehlednost toku dat je možno v grafu použít značení šipkou. Další způsob je znázorněn na obrázku 2.26, princip je stejný jako u větvení, jen zde jedna z větví neobsahuje žádný krok, jen přechod. [9][14]



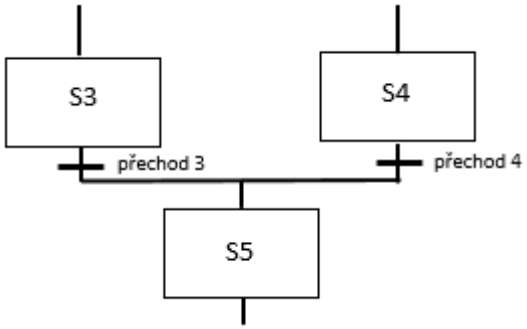
Obrázek 2.19: *Jednoduchá sekvence [9]*



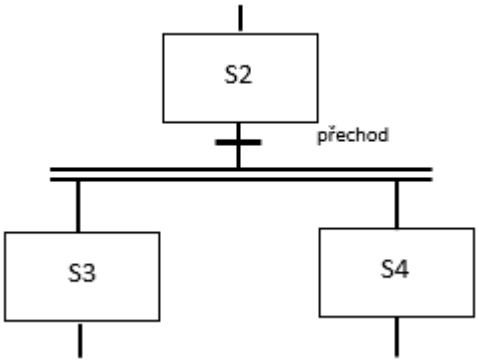
Obrázek 2.20: *Větvení [9]*



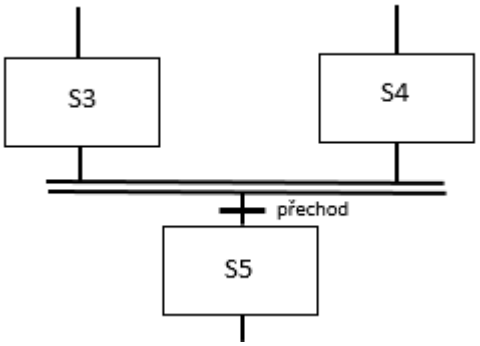
Obrázek 2.21: *Větvení s určenou prioritou [9]*



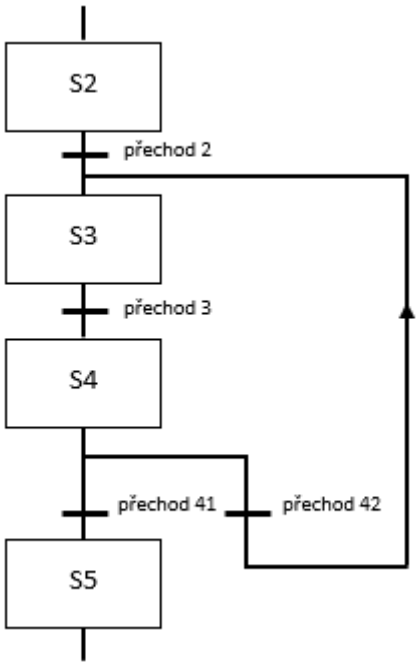
Obrázek 2.22: *Sjednocení [9]*



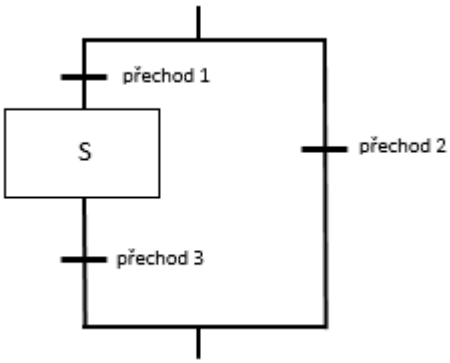
Obrázek 2.23: *Začátek simultánního větvení [9]*



Obrázek 2.24: *Konec simultánního větvení [9]*



Obrázek 2.25: *Sekvenční smyčka [9]*

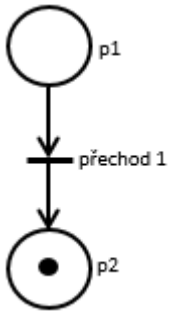


Obrázek 2.26: *Sekvenční přeskočení [9]*

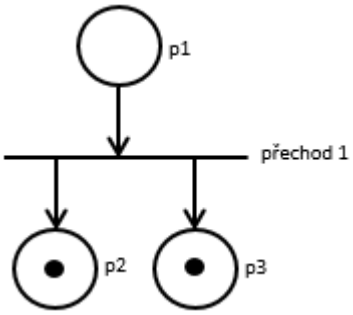
2.3.2 **P/T Petriho sítě**

Dalším nástrojem vhodným pro modelování sekvenčních úloh jsou Petriho sítě. Existuje několik variant Petriho sítí, pro návrh sekvenčního programu pro PLC se jako vhodná varianta jeví P/T (Place/Transitions) Petriho sítě, tato varianta bude v této části popsána. Základ P/T Petriho sítí tvoří místa, přechody a hrany. U P/T Petriho sítí lze najít podobnosti s jazykem SFC. Místa jsou zakreslena jako kružnice a představují stavy programu, podobně jako kroky u SFC. Přechody jsou zakresleny obdélníkem nebo hrubší čarou a jsou to podmínky pro přechod z jednoho místa do místa následujícího. Hrany spojují místa s přechody a jsou znázorněny čárou s šipkou na jejím konci. Místa mohou mít svou kapacitu, což je maximální počet tokenů, které mohou najednou být v daném místě. Tokeny jsou zaznačeny malou vyplněnou kružnicí a říkají, zda byly splněny podmínky. Pokud jsou podmínky splněny, token se nachází v daném místě, pokud ne, místo je prázdné. V případě, že není uvedena kapacita místa, je nekonečná. Hrany mohou mít udánu svou váhu, ta říká, kolik tokenů se přesune při provedení přechodu, pokud není nastavena, je její hodnota jedna. Místa a hrany se během programu musejí střídat, nesmí následovat dva místa ani dvě hrany za sebou. Aby bylo možné provést přechod,

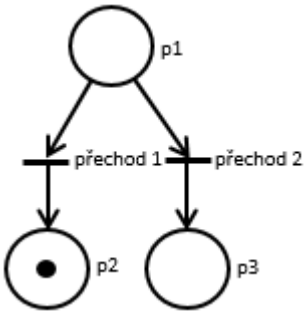
musí platit, že místo před přechodem má minimálně tolik tokenů, jakou hodnotu má násobnost hrany která spojuje dané místo a přechod. Pokud se v místě, které je za přechodem zvýší počet tokenů o hodnotu násobnosti hrany, která do tohoto místa míří, nesmí výsledný počet tokenů převyšovat kapacitu tohoto místa. Poté co se provede přechod, počet tokenu v místech, kdy byly před přechodem, se sníží o násobnost hrany a u míst za přechodem se zase počet tokenů o tuto hodnotu zvýší. [10][11]



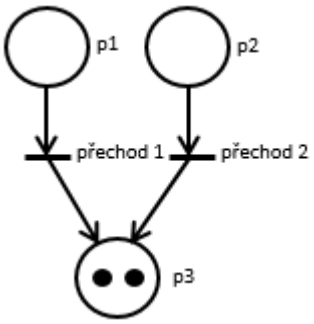
Obrázek 2.27: *Jednoduchá Petriho síť*



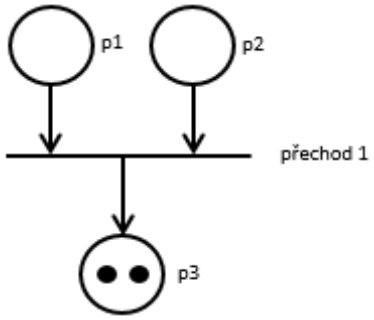
Obrázek 2.28: *Počátek paralelního větvení*



Obrázek 2.29: *Volba jedné ze dvou větví*



Obrázek 2.30: Spojení dvou větví



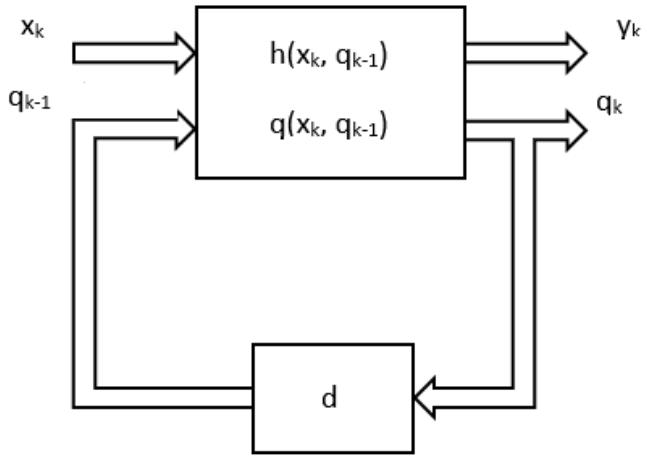
Obrázek 2.31: *Synchronizace*

2.3.3 **Mealyho automat**

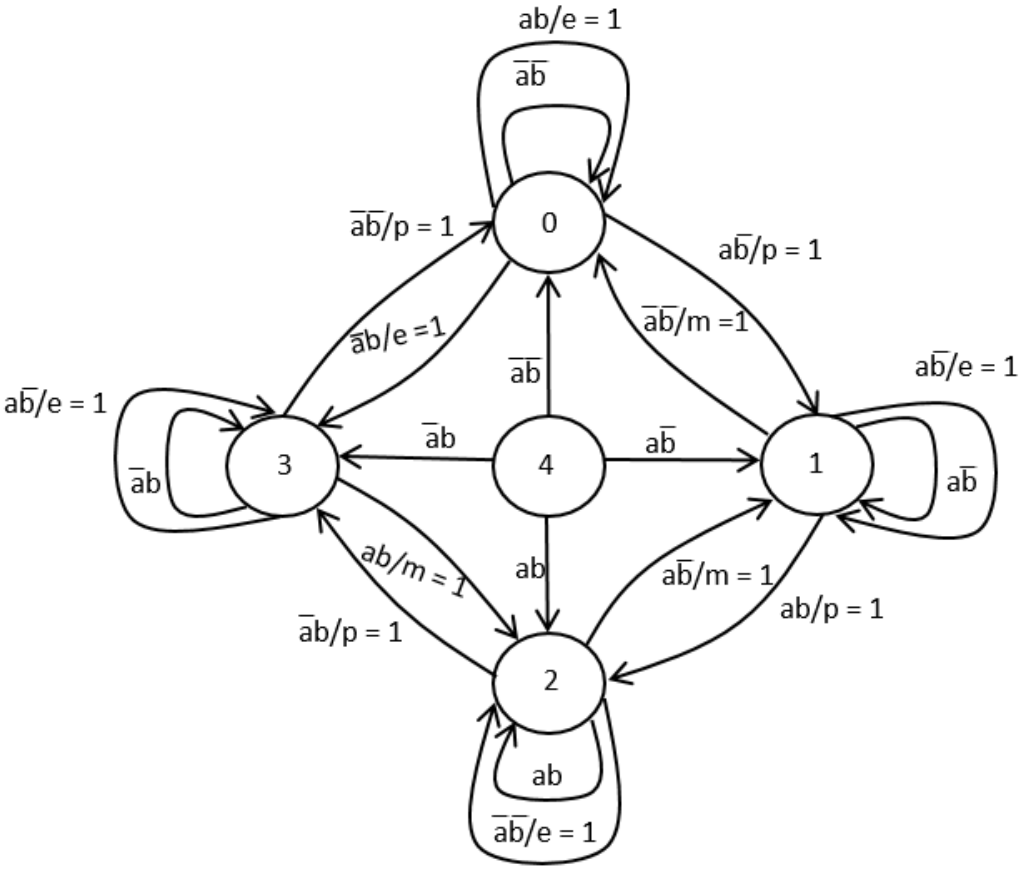
Jednou z dalších možností jak navrhnout sekvenční úlohu je využít metodu konečných automatů, u kterých je typické, že počet vstupních a výstupních kombinací, a také počet stavů, jsou konečné. Mezi konečné automaty řadíme kombinační automaty, automaty se vstupní pamětí, a pro návrh sekvenčních úloh nejvíce využitelné, Mealyho a Mooreův automat, oba tyto automaty mají zpětnou vazbu a díky tomu podávají informaci o stavu systému. Mooreův a Mealyho automat jsou také vzájemné převoditelné a je možné oba automaty kombinovat. Mealyho automat stále vyhodnocuje vstupní a stavové proměnné a podle nich zobrazuje výstupní proměnnou. [4][17]

Mealyho automat lze zobrazit pomocí přechodového diagramu. Jednotlivé stavy jsou v diagramu zakresleny jako kružnice nebo ovály. Stavy bývají očíslovány. Čáry mezi jednotlivými stavy s šipkou na konci směřující k následujícímu stavu představují přechody. Vedle čar znamenající přechody je zapsána podmínka pro přechod, a také se zde uvádí, co se při přechodu stane (např. změna proměnné). Pro oddělení těchto dvou výrazů se mezi nimi píše lomítko. [4]

Na níže uvedeném blokovém schématu představuje x - vstup, y - výstup, q - stav systému, d - zpoždění o jeden výpočetní krok. Přechodová funkce Mealyho automatu je $q_k = g(x_k, q_{k-1})$, výstupní funkce je $y_k = h(x_k, q_{k-1})$. Přechodová i výstupní funkce pracují se vstupní proměnnou x_k a s vnitřní stavovou proměnnou q_{k-1} a po každém přechodu je generována nová výstupní proměnná y_k a také nový krok q_k. [4][17]



Obrázek 2.32: *Blokové schéma Mealyho automatu [4]*

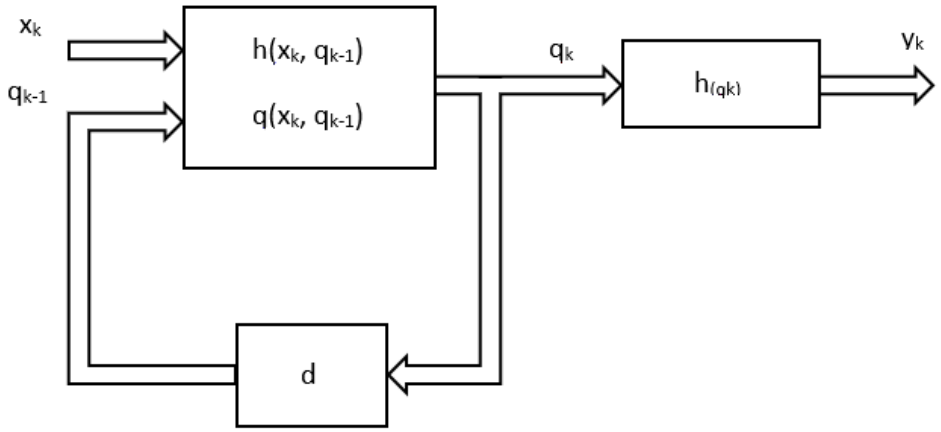


Obrázek 2.33: *Ukázka přechodového diagramu Mealyho automatu [4]*

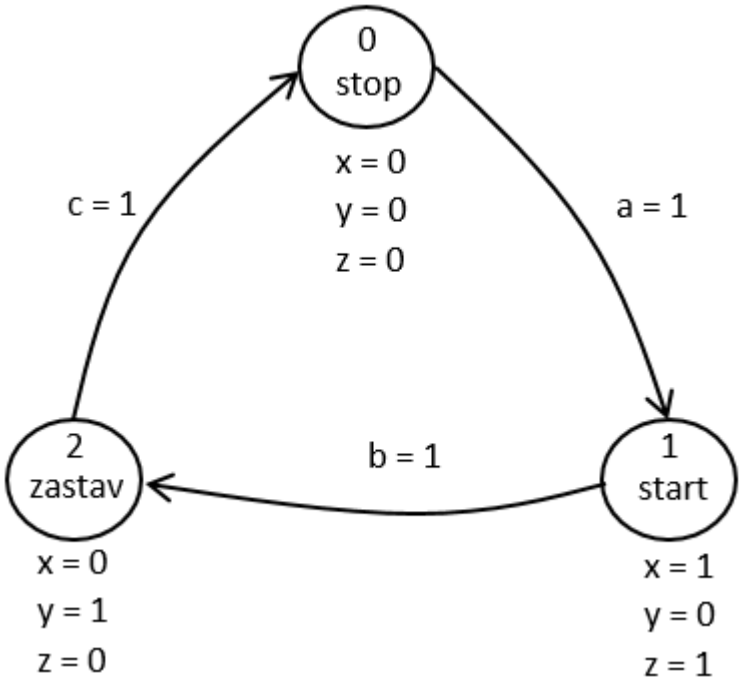
2.3.4 **Mooreův automat**

Mooreův automat, o kterém se někdy hovoří jako o sekvenčním řadiči, se od Mealyho automatu liší především v tom, že u Mooreova automatu závisí výstupní hodnota jen na aktuálním vnitřním stavu, kdežto u Mealyho automatu i na vstupním stavu. Pokud nedojde ke změně stavu, jeho výstupní hodnota zůstává neměnná. Stejně jako Mealyho automat, tak i Mooreův automat lze zobrazit přechodovým diagramem. Pravidla pro kresbu tohoto přechodového diagramu jsou podobné jako u Mealyho automatu, rozdíl je v tom, že u přechodů se píší jen podmínky pro přechod a změna výstupní proměnné se píše u stavů. [4][17]

Na blokovém schématu, které je níže, představuje x - vstup, y - výstup, q - stav systému a symbol d - zpoždění o jeden výpočetní krok. Přechodová funkce Mooreova automatu je $q_k = g(x_k, q_{k-1})$ a pracuje tedy se vstupní proměnnou x_k a také s aktuálním stavem q_{k-1} a určuje informaci o dalším kroku q_k. Výstupní funkce pracuje buďto s hodnotou aktuálního stavu nebo s hodnotou stavu předchozího a určuje výstupní proměnnou. [4][17]



Obrázek 2.34: *Blokové schéma Mooreova automatu [4]*



Obrázek 2.35: *Ukázka přechodového diagramu Mooreova automatu [4]*

3 Návrh nástroje pro zjednodušení tvorby řídicí aplikace

Pomůcka pro tvorbu základu, nebo kompletních řídicích aplikací, jednoduchých sekvenčních úloh byla realizována v tabulkovém procesoru MS Excel 2016. Cílem bylo pokusit se do jisté míry automatizovat a tudíž urychlit a zjednodušit tvorbu řídicí aplikace a následně vhodným způsobem tuto aplikaci nahrát do vybraného nástroje pro programování programovatelných automatů, TIA Portal V13. Při realizaci tohoto nástroje jsem uplatnil jak vlastní znalosti z programování programovatelných automatů, tak jsem také rozšířil své znalosti z používání funkcí tabulkového procesoru MS Excel, a nově jsem se snažil pochopit programování maker v jazyce Visual Basic.

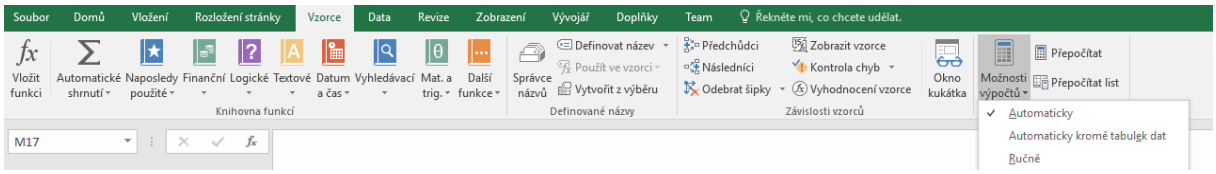
Při tvorbě byla brána inspiraci z programovacího jazyka vytvořeného pro tvorbu sekvenčních úloh, a to jazyka SFC - Sequential Function Chart, který je zanesen v normě IEC 61131-3. V prvé řadě byl vytvořen list, který uživateli umožňuje definovat proměnné a následně je jednoduchým způsobem, pomocí vloženého tlačítka, které je propojeno s vytvořeným makrem, uložit ve vhodném formátu. Po uložení listu s proměnnými je již jednoduché nahrát soubor s příponou .xlsx přímo do nástroje TIA Portal V13. Jako další byl realizován list, který uživateli umožňuje zapsat program pomocí stavů a přechodů. Převod programu do vhodné podoby pro import do nástroje TIA Portal V13 je prováděn v pozadí pomocí maker a funkcí na pomocných listech. Po vygenerování programu pomocí vloženého tlačítka, které je propojeno s makrem, je možné se na programový kód podívat a případně ho upravit, nebo je možno ho ihned jedním z tlačítek uložit, ve vhodném formátu pro zvolenou řadu PLC, s příponou .awl, která umožňuje zdrojový kód poté nahrát do programovacího prostředí TIA Portal V13.

Během zadávání proměnných na listu Zadání proměnných jsou údaje kopírovány a v některých případech i lehce pozměněny, aby s nimi TIA Portal V13 dokázal pracovat, na list PLC Tags. Na listu PLC Tags je pomocí funkcí doplněno před proměnné znak % a slova Ano a Ne jsou změněny na True respektive False. Po stisknutí tlačítka Uložení tabulky se spustí makro UkladaniTagu, díky kterému se všechny proměnné uloží ve vhodném formátu. Poté co uživatel napíše program na listu Zadání, zvolí vhodné hodnoty časovačů na listu Definování časovačů a stiskne tlačítko generování kódu, spustí se makro SpojeniSloupcu, které převede napsaný kód na list PomList, na kterém jsou provedeny téměř všechny úpravy kódu, do dvou sloupců. Kód není makrem nijak pozměněn a je ponechán ve stejné podobě, jako je na listu Zadání, pouze jsou jednotlivé části kódu seřazeny pod sebe do dvou sloupců. Ve sloupci D je kód spojen do jednoho sloupce, převeden téměř do finální podoby, kterou dokáže TIA Portal V13 zpracovat, a také jsou zde nahrazeny některá klíčová slova. Slovo Přechod nahradí NETWORK pro oddělení jednotlivých částí programu a tedy lepší čitelnost kódu, slovo Komentář je nahrazeno TITLE =, aby bylo možné komentář v prostředí TIA Portal V13 zobrazit. Místo Stav proměnné je napsáno A "SX", kde X je číslo aktuálního stavu a to proto, aby mohly být stavy a přechody aktivní jen pokud jsou splněny zadané podmínky. Namísto slova Stav je kromě prvního případu vždy napsáno NETWORK_SX, kde X je číslo aktuálního stavu. Ve sloupci D jsou uvedeny jen typy časovačů a jejich název je uveden ve vedlejším sloupci E pro následné vyfiltrování do kontingenční tabulky. Ve sloupci G jsou pomocí makra Vloz nahrazena klíčová slova NETWORK_SX a typy časovačů víceřádkovým kódem, který je na pomocném listu TimeryNetworky. Místo typu časovače se napíše vhodný kód pro daný typ časovače a místo NETWORK_SX je napsáno R "SM", R "SN", NETWORK, kde M je číslo předchozího stavu, aby byl ve stavu RESET, a N je číslo následujícího stavu, aby byl ve stavu SET. Díky toho je nutné pouze aktivovat první stav a ostatní stavy jsou již aktivovány vždy po splnění podmínky přechodu. NETWORK je zde opět pro oddělení jednotlivých částí kódu a tedy

lepší čitelnosti programu. Nástroj umožňuje vytvořit až 99 stavů a přechodů, avšak není problém toto číslo rozšířit na listu TimeryNetworky. Ve sloupci I jsou do kódu doplněny názvy časovačů a jejich hodnoty, tyto údaje jsou brány z listu Definování časovačů, kde je vytvořena kontingenční tabulka, která zobrazuje všechny časovače, a do vedlejšího sloupce jsou vepisovány hodnoty časovačů. Ve sloupci K jsou pouze doplněny středníky na vhodných místech kódu. Bez středníků by TIA Portal V13 program sice zpracoval, avšak ne zcela správně. Kódy jsou následně kopírovány na listy MainKod1500 a MainKod300. Na těchto listech jsou před kódy umístěny vhodné hlavičky kódu a za kód je pomocí makra VlozKonec vloženo END_ORGANIZATION_BLOCK, což zajistí, že TIA Portal V13 nebude vkládat zbytečné prázdné řádky kódu navíc. Z listu Pomocný list jsou brány údaje do rozevíracích seznamů pro list Zadání a Zadání proměnných. Uložení kódu probíhá pomocí maker UkladaniKodu a UkladaniKodu2, které uloží kódy z listu MainKod1500 nebo MainKod300. Pro větší pohodlí uživatele jsou pomocné listy skryty. Je možné je zobrazit kliknutím na název jakéhokoliv listu pravým tlačítkem myši a zvolením možnosti Zobrazit vyskočí okno s názvy skrytých listů, poté se jen zvolí, který list má být zobrazen. Pokud by pro uživatele byl prostor vyhraněný pro psaní kódu nedostatečný, je možné ho jednoduše rozšířit označením buňky, která je v prostoru pro psaní kódu a chycením pravého spodního rohu roztáhnout tento prostor na požadovanou velikost.

Kód je možno vygenerovat pro PLC řady SIMATIC S7-300, SIMATIC S7-400 a SIMATIC S7-1500. Pro PLC řady SIMATIC S7-1200 není možné kód vygenerovat a to z toho důvodu, že tato řada PLC neumožňuje psát program v textovém jazyce STL. Kód je vygenerován pomocí vloženého tlačítka, které je propojeno s makrem, uložení kódu je zajištěno pomocí dvou vložených tlačítek, která jsou propojena s makry.

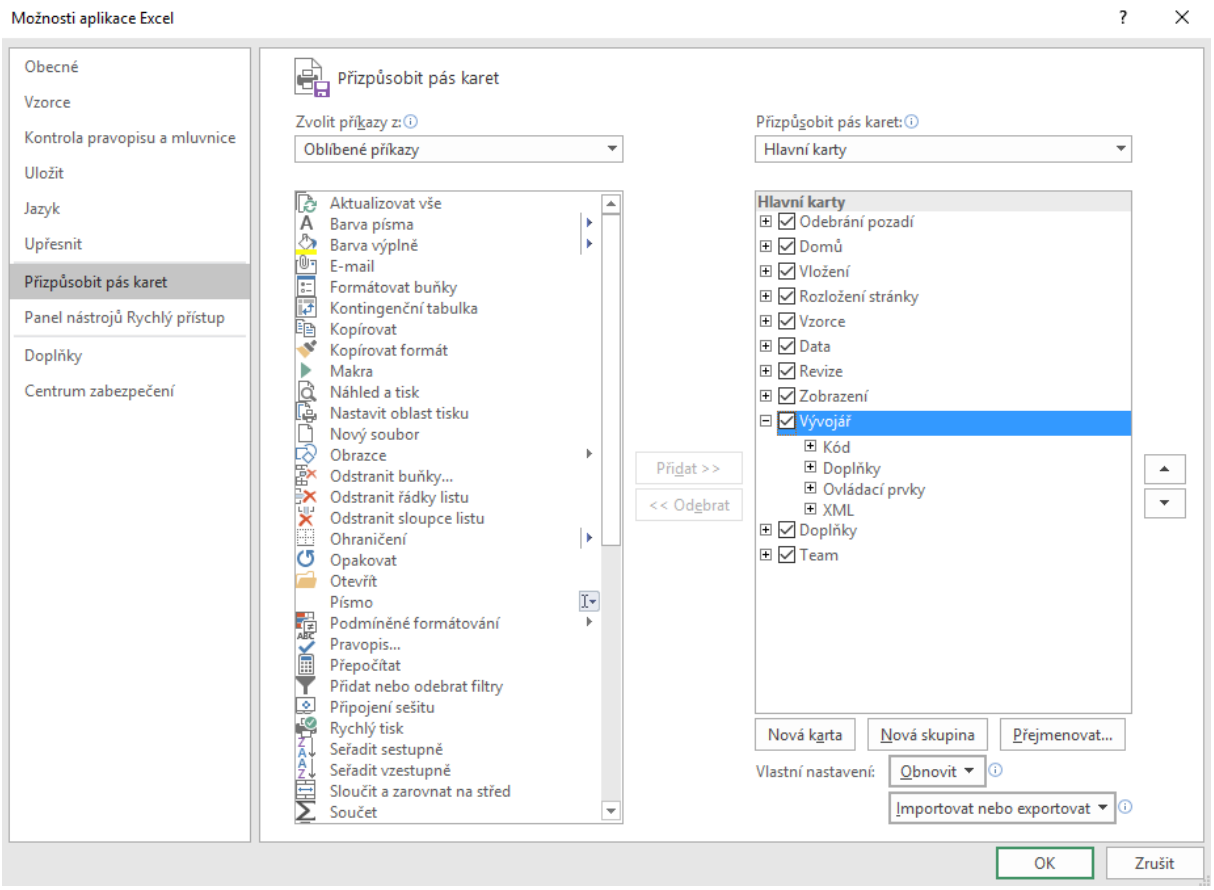
Důležitou podmínkou pro správnou funkci vytvořeného nástroje je mít v Excelu možnost výpočtů nastavenou na „Automaticky“. Tato možnost je sice v MS Excel 2016 nastavena defaultně, avšak pro jistotu je uveden postup, jak automatické přepočítávání vzorců nastavit. V české verzi Excelu po kliknutí na záložku Vzorce v části Výpočet je nutné zvolit Možnosti výpočtů a kliknout na možnost Automaticky.



Obrázek 3.1: Nastavení automatického přepočítávání vzorců

3.1 Vytvoření maker

Převod vytvořeného kódu z listu Zadání do finální podoby, se kterou dokáže pracovat TIA Portal V13 zajišťují jak funkce v Excelu, tak především makra. Aby bylo možné v Excelu makra vytvářet, je nutné nejdříve zobrazit kartu Vývojář, toho lze docílit kliknutím na záložku Soubor, poté kliknutím na Možnosti a zobrazí se okno Možnosti aplikace Excel, zde je nutné kliknout na Přizpůsobit pás karet a v pravé části zaškrtnout možnost Vývojář. Zobrazí se záložka Vývojář a v ní možnost Visual Basic, která umožňuje vytvářet vlastní makra.



Obrázek 3.2: *Zobrazení karty vývojář*

3.2 Definování proměnných

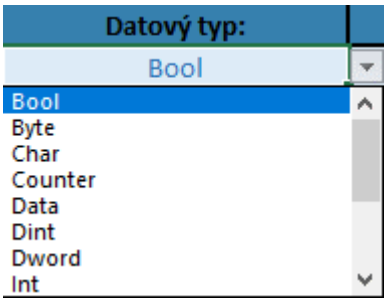
Proměnné lze definovat na listu s názvem Zadání proměnných. Na listu se do tabulky proměnných zadají všechny potřebné údaje. Údaje, u kterých je to možné, lze pro zjednodušení a minimalizaci chyb v zápisu volit z rozevíracích seznamů. Kromě proměnných, které jsou použity pro psaní programu, je nutné také definovat jednotlivé stavy a to jako datový typ bool. Proměnné se automaticky přepíší do vhodného tvaru na pomocném listu PLC Tags pomocí jednoduchých funkcí programu Excel. Časovače pojmenováváme jako TimerX, kde místo X musí být použito jejich pořadové číslo, toto pevně dané pojmenování je zvoleno kvůli možnosti časovače poté vyfiltrovat do kontingenční tabulky, díky které je možné volit jejich hodnoty. Aby bylo možné přecházet mezi stavy, je také nutné definovat jednotlivé stavy, ty se zapisují jako SX, kde místo X je napsáno jejich pořadové číslo. Datový typ stavů volíme Bool. Po zvolení všech potřebných údajů je možné tabulku uložit pomocí tlačítka Uložení tabulky, které se nachází napravo od tabulky. Tlačítku je přiřazeno makro UkladaniTagu, které umožňuje zvolit místo uložení a následně uložit pomocný list PLC Tags s vhodnou příponou. Po kliknutí na tlačítko Uložení tabulky je uživatel vyzván pro zvolení cesty a zadání názvu pro uloženou tabulku. Kliknutím na Uložit je tabulku uložena ve vhodném formátu pro nahrání do programu TIA Portal V13.

Při tvorbě tohoto nástroje bylo důležité zvolit na listu PLC Tags textový formát buněk, protože česká verze MS Excel 2016 vždy při uložení toho listu makrem přepsala buňky True na PRAVDA a buňky False na NEPRAVDA.

Název proměnné:	Cesta:	Datový typ:	Logická adresa:	Komentář:	HMI viditelný:	HMI přístup:
S1	Default tag table	Bool	M0.0	Stav 1	Ano	Ano
S2	Default tag table	Bool	M0.1	Stav2	Ano	Ano
S3	Default tag table	Bool	M0.2	Stav 3	Ano	Ano
S4	Default tag table	Bool	M0.3	Stav 4	Ano	Ano
S5	Default tag table	Bool	M0.4	Stav 5	Ano	Ano
Timer1	Default tag table	Timer	T1	Časovač 1	Ano	Ano
Timer2	Default tag table	Timer	T2	Časovač 2	Ano	Ano
Timer3	Default tag table	Timer	T3	Časovač 3	Ano	Ano
Timer4	Default tag table	Timer	T4	Časovač 4	Ano	Ano
Timer5	Default tag table	Timer	T5	Časovač 5	Ano	Ano
PTL1	Default tag table	Bool	M0.5	Tlačítko pro chodce 1	Ano	Ano
PTL2	Default tag table	Bool	M0.6	Tlačítko pro chodce 2	Ano	Ano
G	Default tag table	Bool	Q0.0	Zelená	Ano	Ano
Y	Default tag table	Bool	Q0.1	Oranžová (žlutá)	Ano	Ano
R	Default tag table	Bool	Q0.2	Červená	Ano	Ano
GP1	Default tag table	Bool	Q0.3	Zelená chodci 1	Ano	Ano
RP1	Default tag table	Bool	Q0.4	Červená chodci 1	Ano	Ano
GP2	Default tag table	Bool	Q0.5	Zelená chodci 2	Ano	Ano
RP2	Default tag table	Bool	Q0.6	Červená chodci 2	Ano	Ano

Uložení tabulky

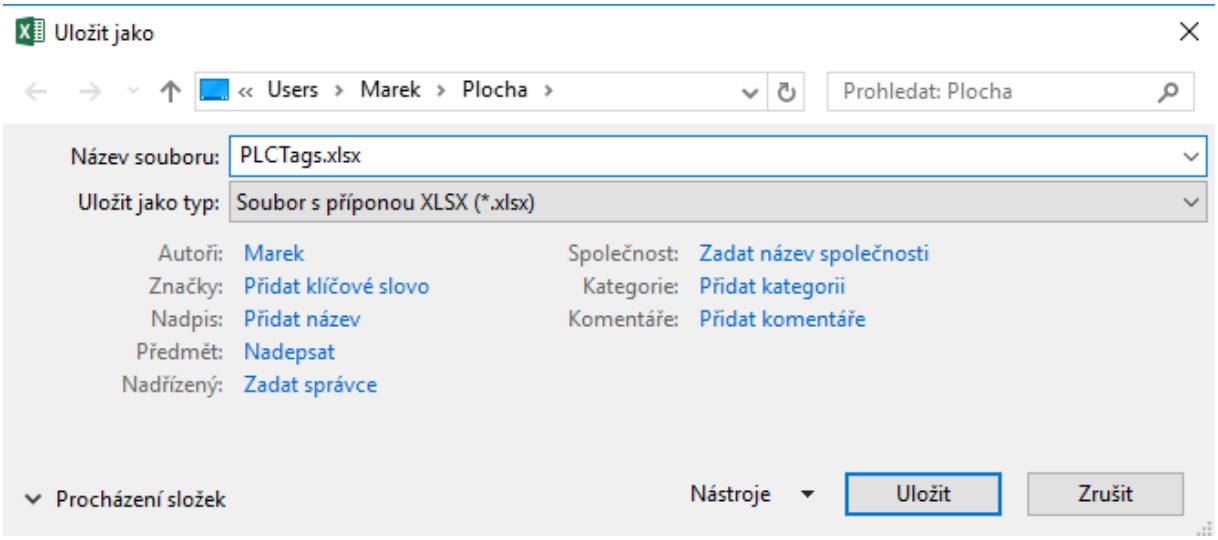
Obrázek 3.3: *Kompletně zadaná tabulka proměnných v tabulkovém procesoru MS Excel*



Obrázek 3.4: *Rozevírací seznam pro zvolení datového typu*

Name	Path	Data Type	Logical Address	Comment	Hmi Visible	Hmi Accessible
S1	Default tag table	Bool	%M0.0	Stav 1	True	True
S2	Default tag table	Bool	%M0.1	Stav2	True	True
S3	Default tag table	Bool	%M0.2	Stav 3	True	True
S4	Default tag table	Bool	%M0.3	Stav 4	True	True
S5	Default tag table	Bool	%M0.4	Stav 5	True	True
PTL	Default tag table	Bool	%I0.0	Pomocné tlačítko	True	True
G	Default tag table	Bool	%Q0.0	Zelená	True	True
Y	Default tag table	Bool	%Q0.1	Oranžová (žlutá)	True	True
R	Default tag table	Bool	%Q0.2	Červená	True	True
Timer1	Default tag table	Timer	%T1	Časovač 1	True	True
Timer2	Default tag table	Timer	%T2	Časovač 2	True	True
Timer3	Default tag table	Timer	%T3	Časovač 3	True	True
Timer4	Default tag table	Timer	%T4	Časovač 4	True	True

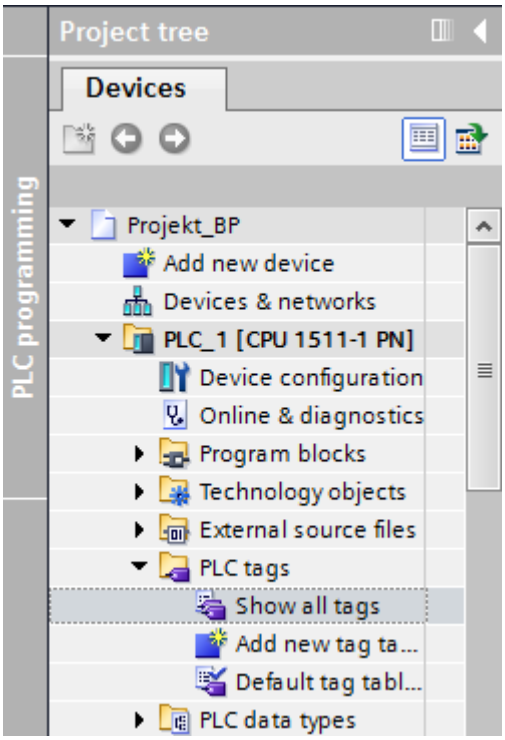
Obrázek 3.5: *Pomocný list PLC Tags s tabulkou proměnných nachystanou na uložení*



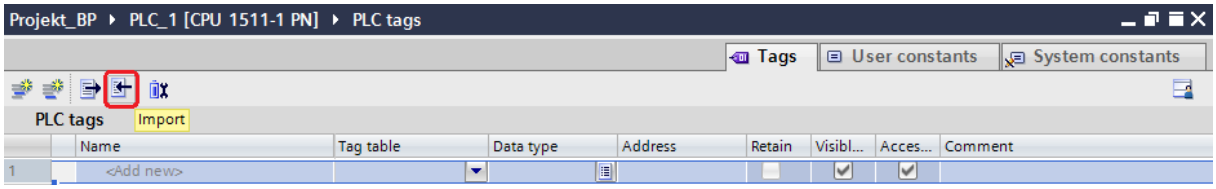
Obrázek 3.6: *Výběr názvu a místa uložení tabulky proměnných*

3.2.1 Nahrání souboru s proměnnými do programu TIA Portal

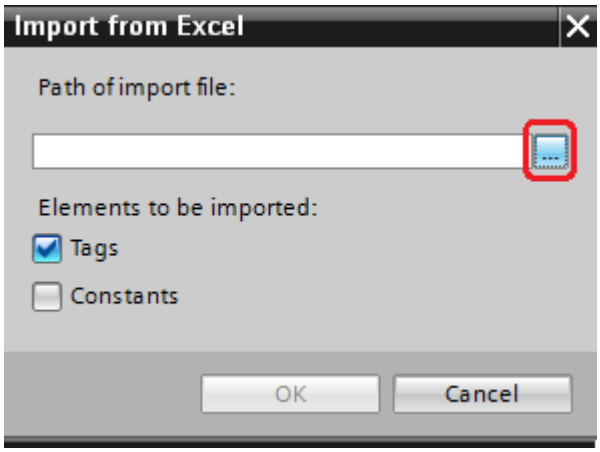
Nahrání tabulky do TIA Portalu V13 se provede tak, že v levé části TIA Portalu V13 se dvojklikem na složku PLC tags složka rozbalí, a dvojklikem na možnost Show all tags zobrazíme tabulku s proměnnýma. Poté se klikne na ikonu Import a zvolí se cesta, kde se nachází soubor s proměnnými, který chce uživatel nahrát. Po zvolení souboru se klikne na Ok, po nahrání vyskočí hláška, že Import je dokončen s upozorněními. Upozornění poukazují na to, že v nahraném souboru chybí verze ID a je předpokládáno, že se jedná o nejnovější verzi. Po kliknutí na Ok jsou všechny proměnné nahrány v programu TIA Portal V13.



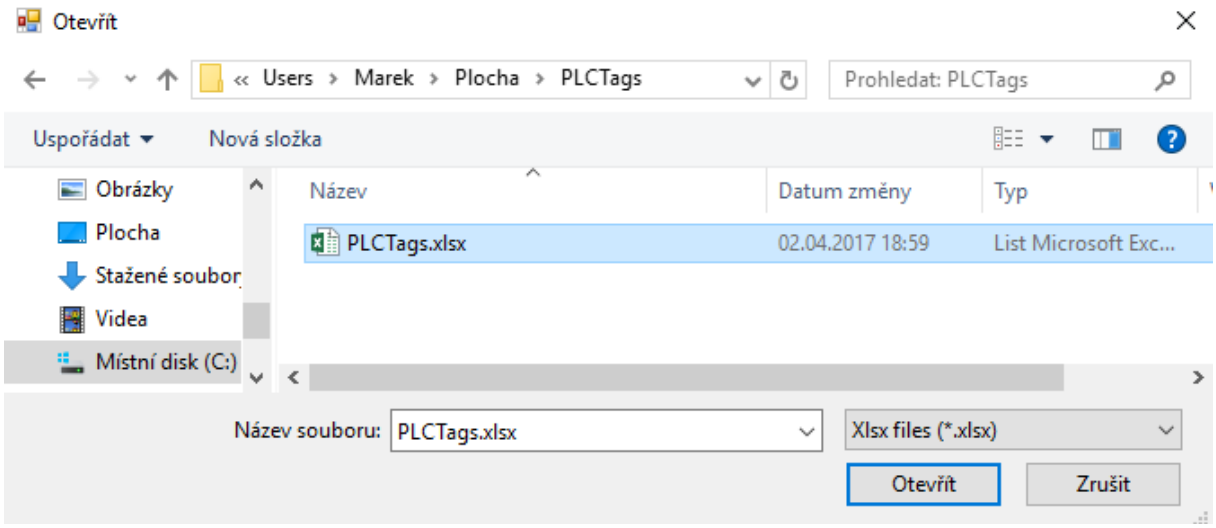
Obrázek 3.7: *Otevření okna se všemi proměnnými*



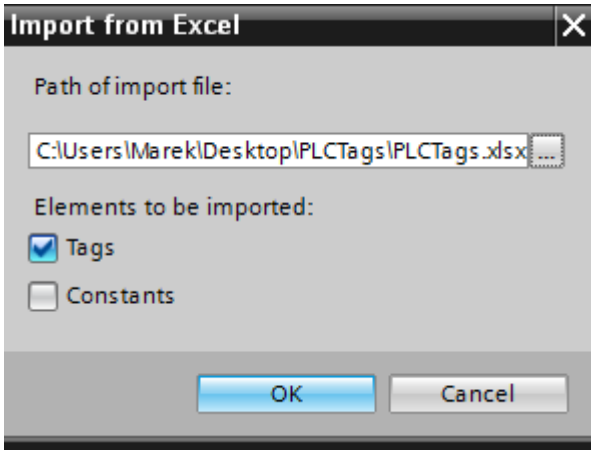
Obrázek 3.8: *Ikona pro import proměnných z Excelu*



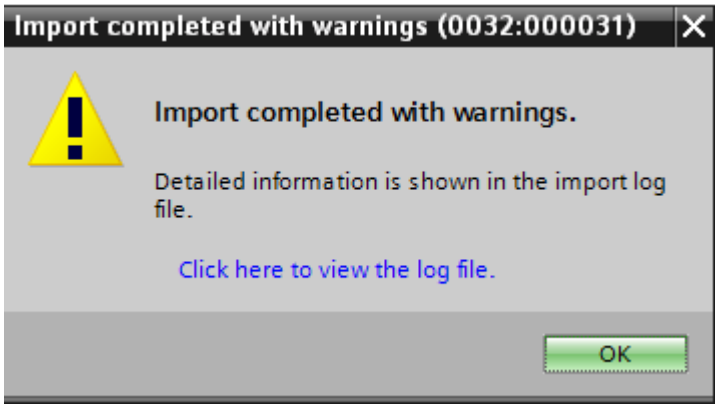
Obrázek 3.9: *Ikona pro zvolení cesty k cílovému souboru*



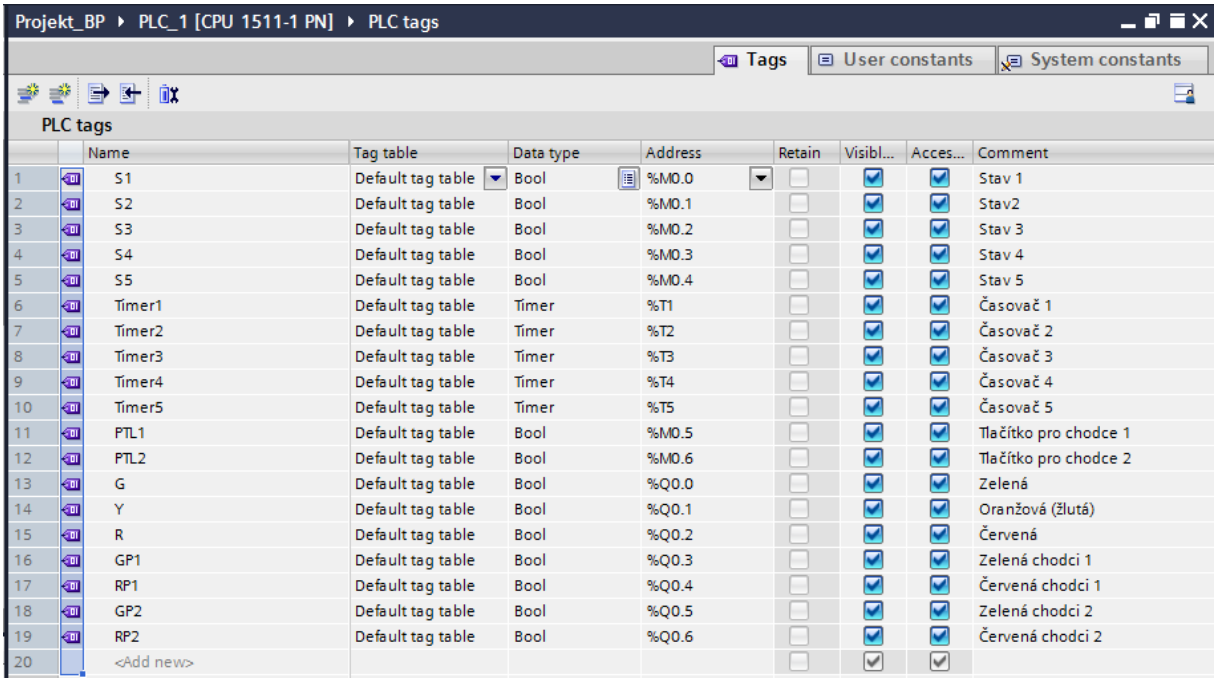
Obrázek 3.10: *Okno pro zvolení cesty k cílovému souboru*



Obrázek 3.11: *Vybraná cesta k cílovému souboru*



Obrázek 3.12: *Hláška s upozorněním*



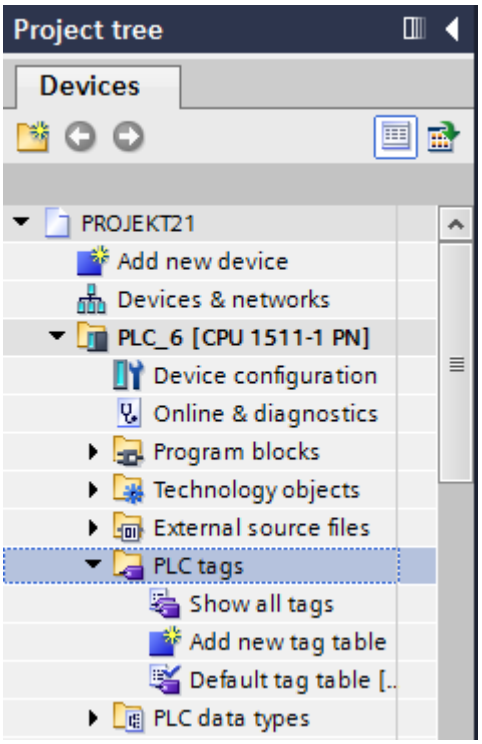
	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...	Comment
1	S1	Default tag table	Bool	%M0.0		☑	☑	Stav 1
2	S2	Default tag table	Bool	%M0.1		☑	☑	Stav2
3	S3	Default tag table	Bool	%M0.2		☑	☑	Stav 3
4	S4	Default tag table	Bool	%M0.3		☑	☑	Stav 4
5	S5	Default tag table	Bool	%M0.4		☑	☑	Stav 5
6	Timer1	Default tag table	Timer	%T1		☑	☑	Časovač 1
7	Timer2	Default tag table	Timer	%T2		☑	☑	Časovač 2
8	Timer3	Default tag table	Timer	%T3		☑	☑	Časovač 3
9	Timer4	Default tag table	Timer	%T4		☑	☑	Časovač 4
10	Timer5	Default tag table	Timer	%T5		☑	☑	Časovač 5
11	PTL1	Default tag table	Bool	%M0.5		☑	☑	Tlačítko pro chodce 1
12	PTL2	Default tag table	Bool	%M0.6		☑	☑	Tlačítko pro chodce 2
13	G	Default tag table	Bool	%Q0.0		☑	☑	Zelená
14	Y	Default tag table	Bool	%Q0.1		☑	☑	Oranžová (žlutá)
15	R	Default tag table	Bool	%Q0.2		☑	☑	Červená
16	GP1	Default tag table	Bool	%Q0.3		☑	☑	Zelená chodci 1
17	RP1	Default tag table	Bool	%Q0.4		☑	☑	Červená chodci 1
18	GP2	Default tag table	Bool	%Q0.5		☑	☑	Zelená chodci 2
19	RP2	Default tag table	Bool	%Q0.6		☑	☑	Červená chodci 2
20	<Add new>					☑	☑	

Obrázek 3.13: *Okno s nahranými proměnnými.*

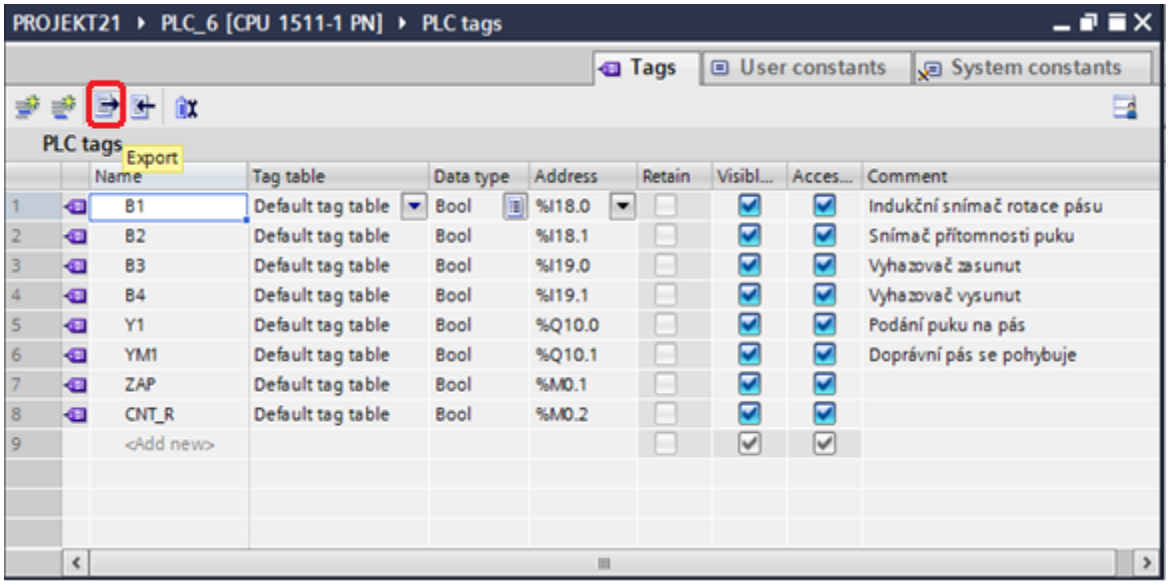
3.2.2 Export proměnných z TIA Portalu

Aby bylo možno dále používat proměnné, které jsou nahrány ve vývojovém prostředí TIA Portal, lze je exportovat do formátu .xlsx a dále je používat v jiných projektech, případně lze soubor s proměnnými jakkoliv modifikovat.

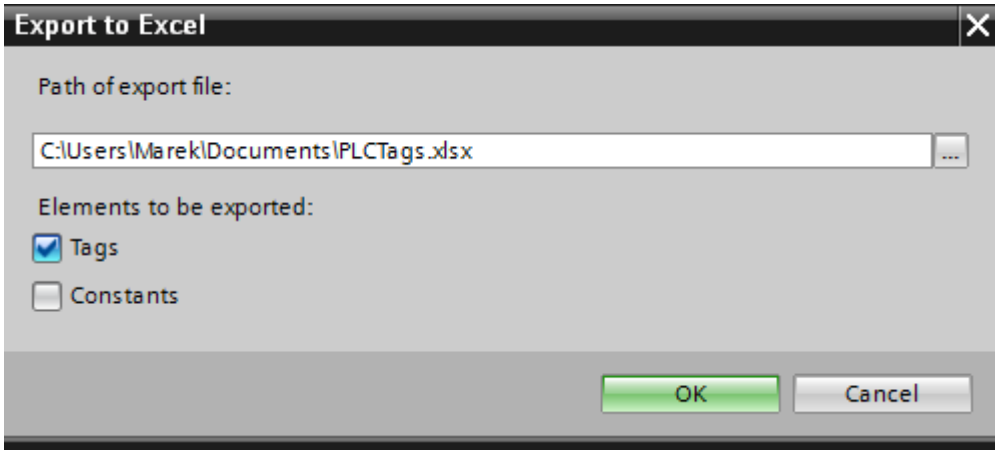
Export tabulky s proměnnými probíhá tak, že se v TIA Portalu v levé části obrazovky v části Project tree dvojklikem otevře složka PLC tags, zvolí se tabulka proměnných, které mají být exportovány, a poté se klikne na ikonu exportu. Dále musí být vybrána cesta, kde bude tabulka uložena a levým tlačítkem myši kliknout na OK. Soubor se uloží s příponou .xlsx a je možné ho otevřít v programu Excel.



Obrázek 3.14: *Složka PLC tags*



Obrázek 3.15: *Tabulka proměnných - ikona pro Export*



Obrázek 3.16: *Výběr cesty pro export tabulky proměnných*

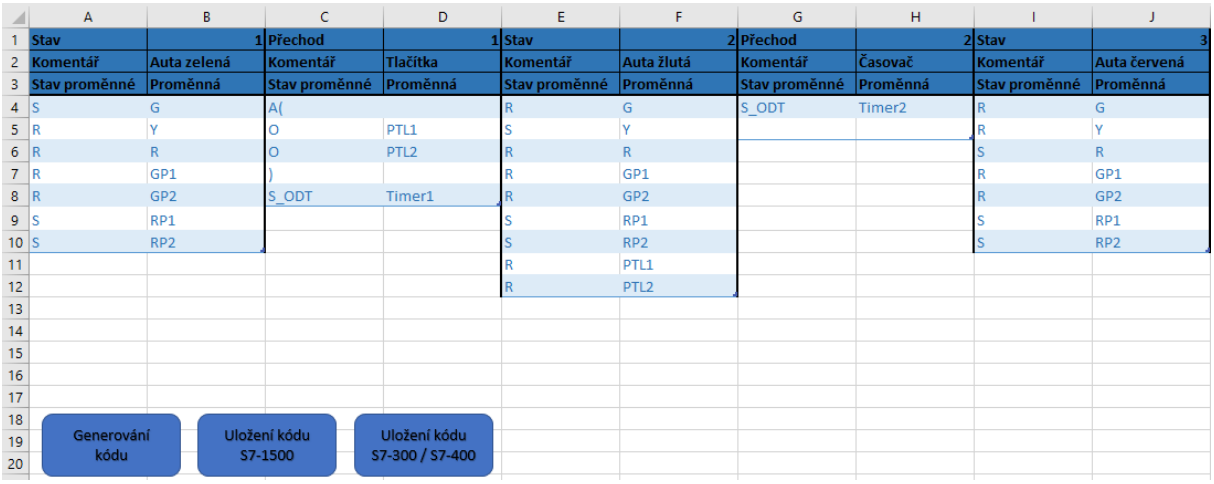
3.2.3 Modifikace souboru s proměnnými

Soubor s proměnnými je případně možno upravit v Excelu 2016 i mimo vytvořený nástroj, je pouze nutné dodržet formát, v jakém se jednotlivé údaje zapisují. Po uložení bude bez problémů možné soubor nahrát do programu TIA Portal.

3.3 Vytvoření řídicí aplikace

Samotnou řídicí aplikaci je možno vytvořit na listu Zadání a při jejím vytváření se musí dodržet některé zásady, podle kterých byl nástroj vytvořen. Na prvním řádku v listu si uživatel volí, zda se bude popisovat stav nebo přechod. Pokud má být do stavu nebo přechodu zapisováno, je vždy nutné na prvním řádku mít zvoleno, zda se používá stav nebo přechod, a také se mezi sebou musí střídat, není tedy možné, aby byly aktivní dva stavy zároveň. Vše je navrženo tak, aby program začínal stavem a po stavu následoval vždy přechod. Vždy se musí také zvolit o který stav v pořadí a také který přechod v pořadí se jedná. Po posledním přechodu si uživatel může zvolit, zda má být program ukončen, nebo chce, aby

se program vrátil do prvního stavu. Pokud je zvolen konec programu, program zůstane v posledním stavu. Pokud uživatel chce, aby nebyl aktivní žádný stav, je nutné poslední stav resetovat. Po nahrání kódu do TIA Portalu je každý stav i každý přechod pro přehlednost rozdělen do jednotlivých networků, ke kterým je možné ve vytvořeném nástroji na druhém řádku přidělit komentář. Do třetího řádku na listu se nic nevypisuje, je vyplňován automaticky a slouží pouze jako pomocný řádek pro uživatele, říká mu, zda má zapisovat proměnnou či stav proměnné. Proměnné i stavy proměnných lze vybírat z rozevíracích seznamů. Ze sekvenčních prvků je možné volit pouze základní časovače, které pokud jsou použity, tak lze definovat jejich hodnoty na listu Definování časovačů. Při definování časovačů na listu Definování časovačů je nutné kliknout pravým tlačítkem myši na nápis časovače v záhlaví tabulky a vybrat možnost Obnovit, čímž se aktualizují data v prvním sloupci. Dále je nutné dodržet pořadí, v jakém jsou časovače na listu Zadání použity, pokud je kontingenční tabulka nezobrazí ve správném pořadí, je možné buňky jednoduše přetáhnout na správnou pozici. Do sloupce Čas se zapíše hodnota času i jednotka a to bez mezer. Pokud je celý program vytvořen, lze ho pomocí tlačítka Generování kódu vygenerovat v podobě, kterou dokáže TIA Portal V13 přečíst. Kód se vygeneruje na list MainKod300 pro PLC řady SIMATIC S7-300 a SIMATIC S7-400 a na list MainKod1500 pro PLC řady SIMATIC S7-1500. Na obou listech lze správnost kódu zkontrolovat. Na listu Zadání jsou dva tlačítka pro uložení kódu, tlačítko s nápisem Uložení kódu S7-1500 pro uložení kódu ve formátu pro PLC řady SIMATIC S7-1500 a tlačítko s nápisem Uložení kódu S7-300 / S7-400 pro uložení kódu ve formátu pro PLC řady SIMATIC S7-300 a S7-400. Kliknutím na jedno z tlačítek pro uložení kódu je uložen vygenerovaný kód ve formátu s koncovkou .awl. Kód z listu Zadání byl do potřebné formy převeden jak pomocí funkcí v Excelu, tak také především pomocí maker.

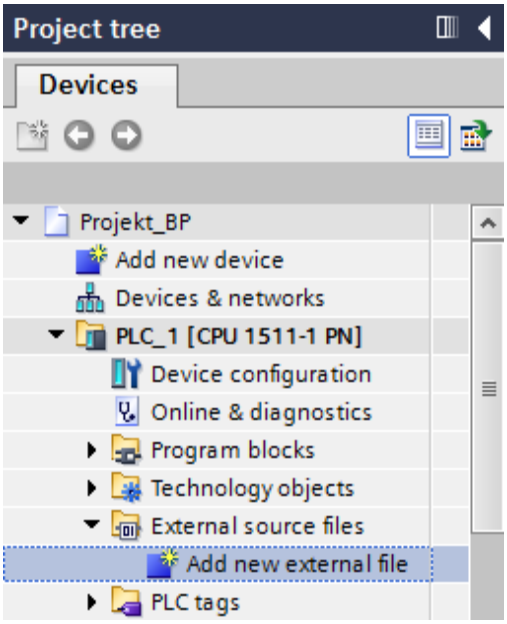


	A	B	C	D	E	F	G	H	I	J
1	Stav	1	Přechod	1	Stav	2	Přechod	2	Stav	3
2	Komentář	Auta zelená	Komentář	Tlačítka	Komentář	Auta žlutá	Komentář	Časovač	Komentář	Auta červená
3	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná
4	S	G	A(		R	G	S_ODT	Timer2	R	G
5	R	Y	O	PTL1	S	Y			R	Y
6	R	R	O	PTL2	R	R			S	R
7	R	GP1	)		R	GP1			R	GP1
8	R	GP2	S_ODT	Timer1	R	GP2			R	GP2
9	S	RP1			S	RP1			S	RP1
10	S	RP2			S	RP2			S	RP2
11					R	PTL1				
12					R	PTL2				

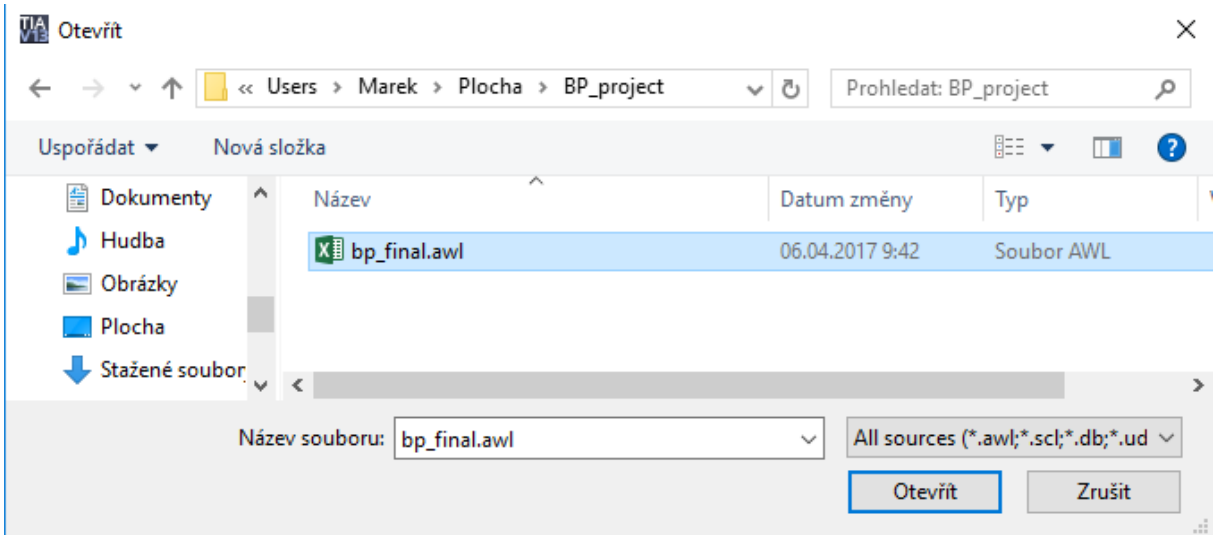
Obrázek 3.17: *Část listu Zadání pro tvorbu kódu*

3.3.1 Nahrání souboru s kódem do programu TIA Portal

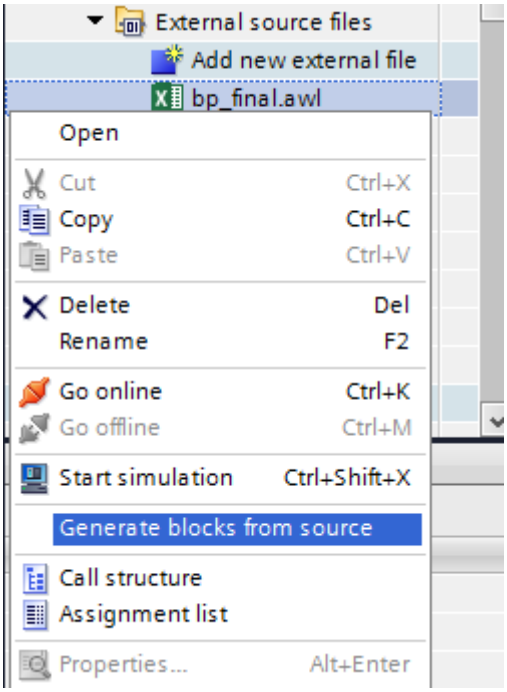
Poté, co se do TIA Portalu nahrál soubor s proměnnými, lze nahrát i soubor se samotným kódem. Soubor se do prostředí TIA Portal nahraje tak, že se dvojklikem rozbalí složka External source files a vybere se možnost Add new external file. Poté se zvolí cesta k souboru s kódem a klikne se na možnost otevřít. Soubor se nahraje do TIA Portalu a poté je nutné na něj kliknout pravým tlačítkem myši a vybrat možnost Generate blocks from source. Pokud je vše uděláno správně, tak vygenerovaný programový blok je k nalezení v části Program blocks.



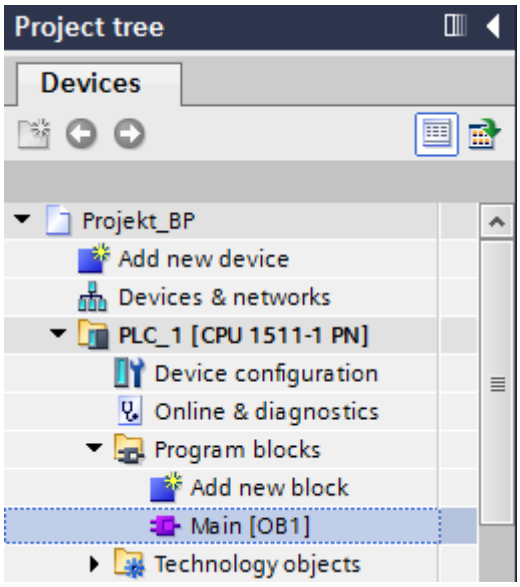
Obrázek 3.18: *Ikona pro přidání nového externího souboru*



Obrázek 3.19: *Zvolení cesty k souboru*



Obrázek 3.20: *Generování programového bloku ze zdrojového souboru*



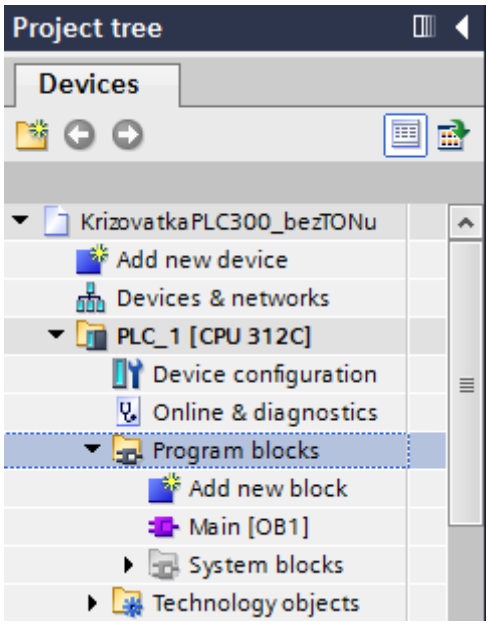
Obrázek 3.21: *Vygenerovaný blok*

3.3.2 Export programových bloků z TIA Portalu

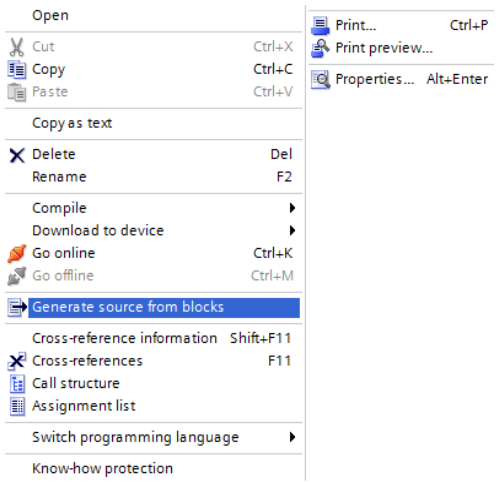
Aby bylo možno dále používat již vytvořený programový blok, který je nahraný ve vývojovém prostředí TIA Portal, lze ho z prostředí TIA Portal exportovat a dále ho použít v jiném programu, případně lze tento soubor upravit. Export je však možný pouze, pokud je programový blok napsán jedním z textových programovacích jazyků, případně lze programový blok do textového jazyka převést.

Export programových bloků probíhá obdobně jako export tabulky proměnných. V části Project tree se otevře složka Program blocks, zvolí se, který blok má být exportován a poté se na něj klikne pravým tlačítkem myši. Dále se zvolí možnost Generate source from blocks a vybere se cesta,

kde se soubor uloží. Soubory se uloží s koncovkou .awl nebo .scl, podle toho v jakém jazyce je daný blok napsán a je možno jej otevřít v programu Excel.



Obrázek 3.22: Složka Program blocks



Obrázek 3.23: *Generování kódu z bloku*

3.3.3 Modifikace souboru s kódem

Při pokusech o ruční uložení souboru v MS Excel 2016 se následně nepovedlo soubor nahrát do TIA Portalu V13 a ani některé pokusy o ukládání pomocí makra nebyly úspěšné, přesto, že soubor měl jak správnou koncovku, tak i správný tvar kódu. Zároveň se nepovedlo v Excelu 2016 upravit vygenerovaný soubor s koncovkou .awl, protože po opětovném uložení soubor sice bylo možné nahrát do TIA Portalu V13, avšak TIA Portal V13 již nebyl schopný tento upravený soubor převést na programový blok. Je to způsobeno tím, že Excel 2016 soubor s koncovkou .awl po modifikaci mírně upraví a kód již poté není v takovém formátu, aby z něj TIA Portal dokázal vygenerovat programový

blok. Je nutné případné úpravy provádět buďto ve vytvořeném nástroji v Excelu a následně soubor uložit pomocí makra, nebo lze soubor otevřít v programech Poznámkový blok a WordPad a úpravy provést tam.

4 Návrh a realizace řídicí aplikace

Celý nástroj vytvořený v Excelu byl otestován při tvorbě jednoduché sekvenční řídicí aplikace, ta byla nejdříve navrhnuta graficky pomocí SFC a následně pomocí vytvořeného nástroje v Excelu realizována. Funkčnost poté byla otestována jak na reálném PLC, tak také pomocí simulačního nástroje S7-PLCSIM.

4.1 Zadání sekvenční úlohy

Pro otestování efektivity a funkčnosti vytvořeného nástroje v programu Excel byla zvolena úloha, která simuluje dopravní semafor se třemi světelnými signály a dva semafory pro chodce, každý se dvěma světelnými signály, dále jsou v úloze využity dva tlačítka pro chodce.

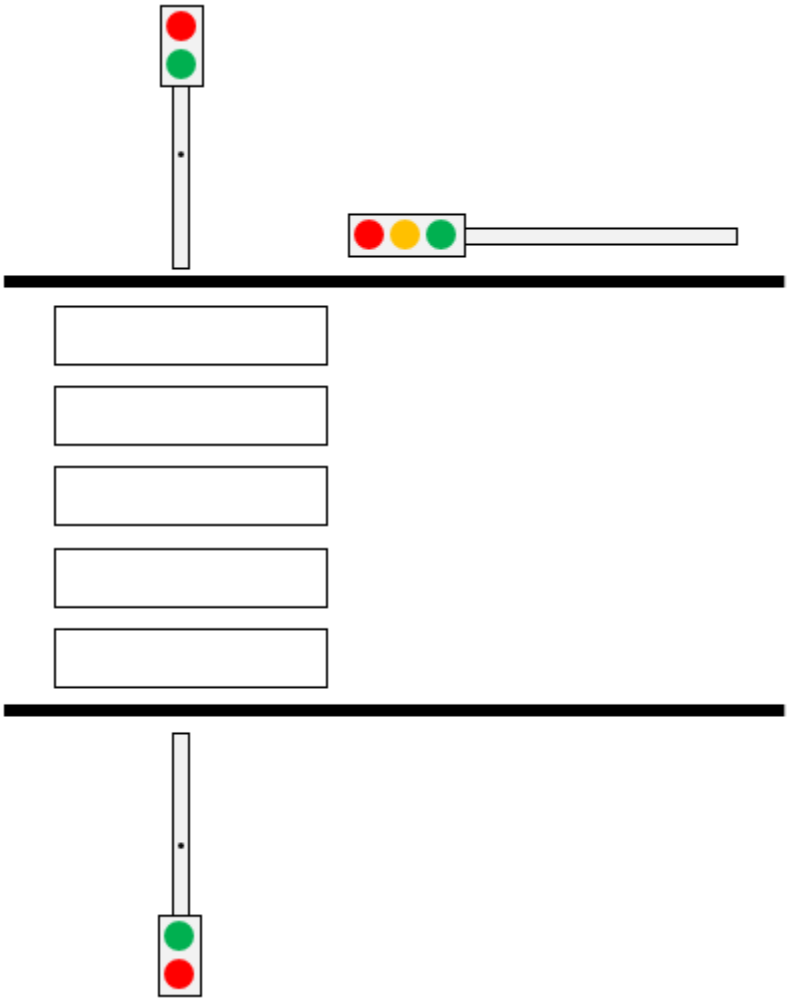
Proměnné G, Y a R představují zelenou, žlutou a červenou barvu na dopravním semaforu, proměnné GP1, GP2 , RP1 a RP2 představují zelenou a červenou barvu na semaforech pro chodce, proměnné PTL1 a PTL2 představují tlačítka pro chodce.

V prvním stavu svítí na semaforu zelené světlo, je tedy aktivní proměnná G a na semaforech pro chodce svítí červená, aktivní jsou proměnné RP1 a RP2. pokud je stisknuto tlačítko a je tedy aktivní PTL1 nebo PTL2, po 5ti sekundách přejde program do stavu 2, zhasne zelené světlo a rozsvítí se na dopravním semaforu světlo žluté barvy, aktivuje se proměnná Y. Po uplynutí 3 sekund se program dostane do stavu 3, zhasne žluté světlo a rozsvítí se červená, aktivní je proměnná R. Po uplynutí 2 sekund je aktivní stav 4, světlo na dopravním semaforu nezmění svou barvu, avšak na semaforech pro chodce se rozsvítí zelená a zhasne červená, aktivují se tedy proměnné GP1 a GP2. Po 20ti sekundách je aktivní stav 5 a na semaforech pro chodce zhasnou zelená světla a opět se aktivují světla červené barvy, zároveň se na dopravním semaforu rozsvítí žluté světlo, které je aktivní zároveň s červeným světlem a signalizuje, že za chvíli se na semaforu rozsvítí zelená. Po uplynutí 2 sekund svítí na dopravním semaforu zelená, a program se opět dostane do prvního stavu.

Tabulka 4.1: Tabulka proměnných

Název	Datový typ	Adresa	Komentář
S1	Bool	%M0.0	Stav 1
S2	Bool	%M0.1	Stav2
S3	Bool	%M0.2	Stav 3
S4	Bool	%M0.3	Stav 4
S5	Bool	%M0.4	Stav 5
Timer1	Timer	%T1	Časovač 1
Timer2	Timer	%T2	Časovač 2
Timer3	Timer	%T3	Časovač 3
Timer4	Timer	%T4	Časovač 4
Timer5	Timer	%T5	Časovač 5
PTL1	Bool	%M0.5	Tlačítko pro chodce 1
PTL2	Bool	%M0.6	Tlačítko pro chodce 2
G	Bool	%Q0.0	Zelená
Y	Bool	%Q0.1	Oranžová (žlutá)
R	Bool	%Q0.2	Červená
GP1	Bool	%Q0.3	Zelená chodci 1
RP1	Bool	%Q0.4	Červená chodci 1
GP2	Bool	%Q0.5	Zelená chodci 2
RP2	Bool	%Q0.6	Červená chodci 2

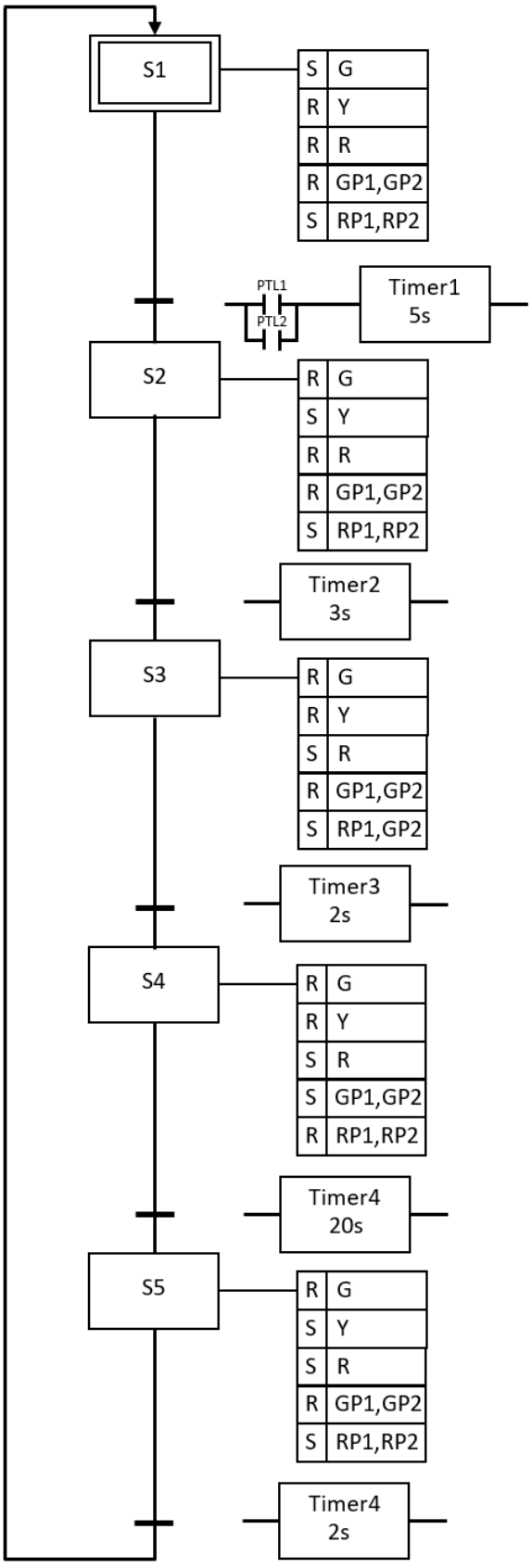
4.2 Grafické znázornění úlohy



Obrázek 4.1: Grafické znázornění úlohy

4.3 Návrh úlohy

Před samotným napsáním programu pomocí vytvořené aplikace v programu Excel byla celá úlohu nejprve graficky znázorněna pomocí SFC diagramu. Díky vytvoření tohoto diagramu je možné postupovat systematicky a vytvoření této sekvenční úlohy je mnohem jednodušší, než kdyby byl celý kód psán z hlavy.



Obrázek 4.2: *SFC diagram zvolené úlohy*

4.4 Vytvoření úlohy

Úloha byla vytvořena pomocí nástroje, vytvořeného v programu Excel. Nejdříve se na listu Zadání proměnných definovaly proměnné, časovače a také stavy tak, jak je vysvětleno v kapitole 3.1. Soubor byl poté pomocí tlačítka Uložení tabulky uložen na zvolené místo.

Název proměnné:	Cesta:	Datový typ:	Logická adresa:	Komentář:	HMI viditelný:	HMI přístup:
S1	Default tag table	Bool	M0.0	Stav 1	Ano	Ano
S2	Default tag table	Bool	M0.1	Stav2	Ano	Ano
S3	Default tag table	Bool	M0.2	Stav 3	Ano	Ano
S4	Default tag table	Bool	M0.3	Stav 4	Ano	Ano
S5	Default tag table	Bool	M0.4	Stav 5	Ano	Ano
Timer1	Default tag table	Timer	T1	Časovač 1	Ano	Ano
Timer2	Default tag table	Timer	T2	Časovač 2	Ano	Ano
Timer3	Default tag table	Timer	T3	Časovač 3	Ano	Ano
Timer4	Default tag table	Timer	T4	Časovač 4	Ano	Ano
Timer5	Default tag table	Timer	T5	Časovač 5	Ano	Ano
PTL1	Default tag table	Bool	M0.5	Tlačítko pro chodce 1	Ano	Ano
PTL2	Default tag table	Bool	M0.6	Tlačítko pro chodce 2	Ano	Ano
G	Default tag table	Bool	Q0.0	Zelená	Ano	Ano
Y	Default tag table	Bool	Q0.1	Oranžová (žlutá)	Ano	Ano
R	Default tag table	Bool	Q0.2	Červená	Ano	Ano
GP1	Default tag table	Bool	Q0.3	Zelená chodci 1	Ano	Ano
RP1	Default tag table	Bool	Q0.4	Červená chodci 1	Ano	Ano
GP2	Default tag table	Bool	Q0.5	Zelená chodci 2	Ano	Ano
RP2	Default tag table	Bool	Q0.6	Červená chodci 2	Ano	Ano

Uložení tabulky

Obrázek 4.3: *Tabulka proměnných na listu Zadání proměnných*

Po vytvoření souboru s proměnnými byl na listu Zadání vytvořen samotný program pro semafory. Je nutné brát v potaz, že v každém stavu a přechodu je vždy automaticky zapsáno, že aby stav/přechod mohl proběhnout, tak musí být daný stav aktivní (programově A "Sx", kde x je číslo aktuálního stavu), proto pokud bude nutné zapsat například logickou funkci OR, je nutné ji zapsat tak, jako je zobrazeno v přechodu 1. Po zapsání programu můžeme přejít na list Definování časovačů a zvolit hodnoty časovačů.

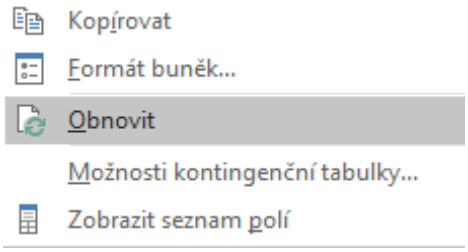
Stav	1	Přechod	1	Stav	2	Přechod	2	Stav	3	Přechod	3
Komentář	Auta zelená	Komentář	Tlačítka	Komentář	Auta žlutá	Komentář	Časovač	Komentář	Auta červená	Komentář	Časovač
Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná
S	G	A(		R	G	S_ODT	Timer2	R	G	S_ODT	Timer3
R	Y	O	PTL1	S	Y			R	Y		
R	R	O	PTL2	R	R			S	R		
R	GP1	)		R	GP1			R	GP1		
R	GP2	S_ODT	Timer1	R	GP2			R	GP2		
S	RP1			S	RP1			S	RP1		
S	RP2			S	RP2			S	RP2		
				R	PTL1						
				R	PTL2						

Obrázek 4.4: *První část kódu na listu Zadání*

Stav	4	Přechod	4	Stav	5	Přechod	5	Přejít do stavu 1	
Komentář	Chodci zelená	Komentář	Časovač	Komentář	Auta žluta + červe	Komentář	Časovač		
Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Stav proměnné	Proměnná	Ze stavu	5
R	G	S_ODT	Timer4	R	G	S_ODT	Timer5		
R	Y			S	Y				
S	R			S	R				
S	GP1			R	GP1				
S	GP2			R	GP2				
R	RP1			S	RP1				
R	RP2			S	RP2				

Obrázek 4.5: *Druhá část kódu na listu Zadání*

Jako první krok při definování hodnot časovačů je důležité kliknout pravým tlačítkem myši na nápis časovače a zvolit možnost obnovit, tabulka s časovači se poté aktualizuje a jsou v ní všechny časovače, které byly použity.



Obrázek 4.6: *Obnovení tabulky s časovači*

Časovače jsou seřazeny od nejmenšího čísla po největší, pokud by byly na listu Zadání použity v jiném pořadí, bylo by nutné změnit jejich pořadí v této tabulce ručně, například přetažením. Pokud jsou použity v pořadí od nejmenšího čísla po největší, stačí vyplnit hodnoty do sloupce čas.

Časovače	Čas
Timer1	5s
Timer2	3s
Timer3	2s
Timer4	20s
Timer5	2s

Obrázek 4.7: *Tabulka s časovači*

Po zvolení hodnot časovačů byl kliknutím na listu Zadání na tlačítko Generování kódu vygenerován kód na listu MainKod1500.

```
ORGANIZATION_BLOCK "Main"
TITLE = "Main Program Sweep (Cycle)"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1

BEGIN
NETWORK
   A "S1";
   S "G";
   R "Y";
   R "R";
   R "GP1";
   R "GP2";
   S "RP1";
   S "RP2";
```

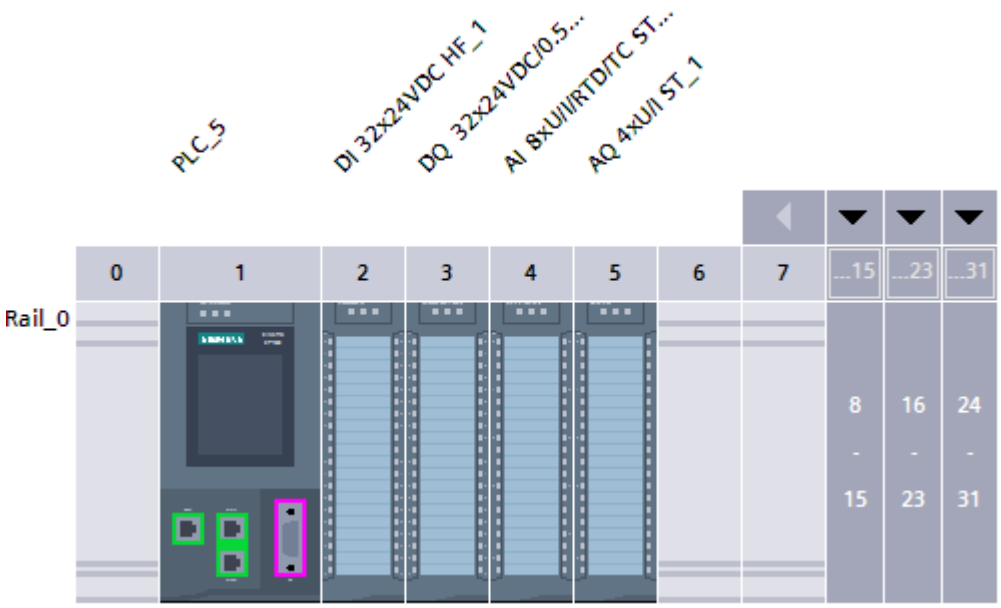
Obrázek 4.8: *Část vygenerovaného kódu pro PLC řady SIMATIC S7-1500*

Kliknutím na tlačítko Uložení kódu na listu Zadání a zvolení cesty se soubor s kódem uložil na zvolené místo. Poté byly oba soubory nahrány do vývojového prostředí TIA Portal V13.

4.5 Otestování úlohy

Úloha byla otestována jak na reálném PLC řady SIMATIC S7-1500, tak i pomocí simulačního softwaru S7-PLCSIM na automatech řady SIMATIC S7-300, SIMATIC S7-400 a SIMATIC S7-1500. V této kapitole bude detailněji popsáno testování na reálném PLC řady SIMATIC S7-1500.

Po vytvoření projektu v programu TIA Portal V13 jsem zvolil vhodnou HW konfiguraci.



Obrázek 4.9: *HW konfigurace*

V dalším kroku byla do TIA Portalu nahrána tabulku proměnných podle postupů zmíněných dříve.

PLC tags

	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...	Comment
1	S1	Default tag table	Bool	%M0.0		☑	☑	Stav 1
2	S2	Default tag table	Bool	%M0.1		☑	☑	Stav2
3	S3	Default tag table	Bool	%M0.2		☑	☑	Stav 3
4	S4	Default tag table	Bool	%M0.3		☑	☑	Stav 4
5	S5	Default tag table	Bool	%M0.4		☑	☑	Stav 5
6	Timer1	Default tag table	Timer	%T1		☑	☑	Časovač 1
7	Timer2	Default tag table	Timer	%T2		☑	☑	Časovač 2
8	Timer3	Default tag table	Timer	%T3		☑	☑	Časovač 3
9	Timer4	Default tag table	Timer	%T4		☑	☑	Časovač 4
10	Timer5	Default tag table	Timer	%T5		☑	☑	Časovač 5
11	PTL1	Default tag table	Bool	%M0.5		☑	☑	Tlačítko pro chodce 1
12	PTL2	Default tag table	Bool	%M0.6		☑	☑	Tlačítko pro chodce 2
13	G	Default tag table	Bool	%Q0.0		☑	☑	Zelená
14	Y	Default tag table	Bool	%Q0.1		☑	☑	Oranžová (žlutá)
15	R	Default tag table	Bool	%Q0.2		☑	☑	Červená
16	GP1	Default tag table	Bool	%Q0.3		☑	☑	Zelená chodci 1
17	RP1	Default tag table	Bool	%Q0.4		☑	☑	Červená chodci 1
18	GP2	Default tag table	Bool	%Q0.5		☑	☑	Zelená chodci 2
19	RP2	Default tag table	Bool	%Q0.6		☑	☑	Červená chodci 2
20	<Add new>					☑	☑	

Obrázek 4.10: *Nahraná tabulka proměnných*

Poté byl nahrán soubor s kódem a z něj byl podle dříve uvedeného postupu vytvořen programový blok.

Network 1: Auta zelená

Comment

```
1   A     "S1"      %M0.0
2   S     "G"       %Q0.0
3   R     "Y"       %Q0.1
4   R     "R"       %Q0.2
5   R     "GP1"     %Q0.3
6   R     "GP2"     %Q0.5
7   S     "RP1"     %Q0.4
8   S     "RP2"     %Q0.6
```

Obrázek 4.11: *Ukázka části kódu*

Projekt se po kompilaci nahrál do zařízení a jeho funkčnost byla ověřena pomocí monitoringu.

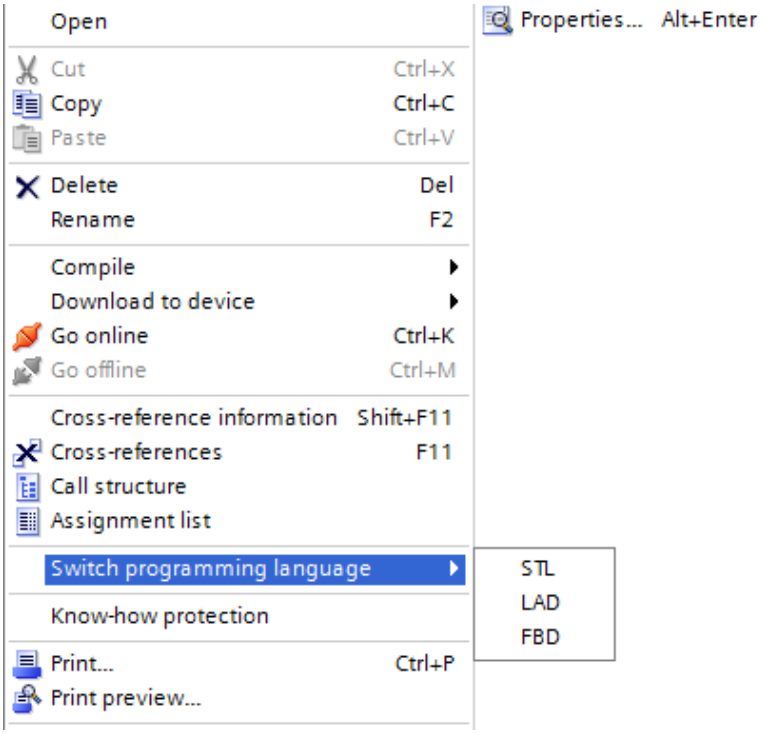
Network 2: Tlaèítka

Comment

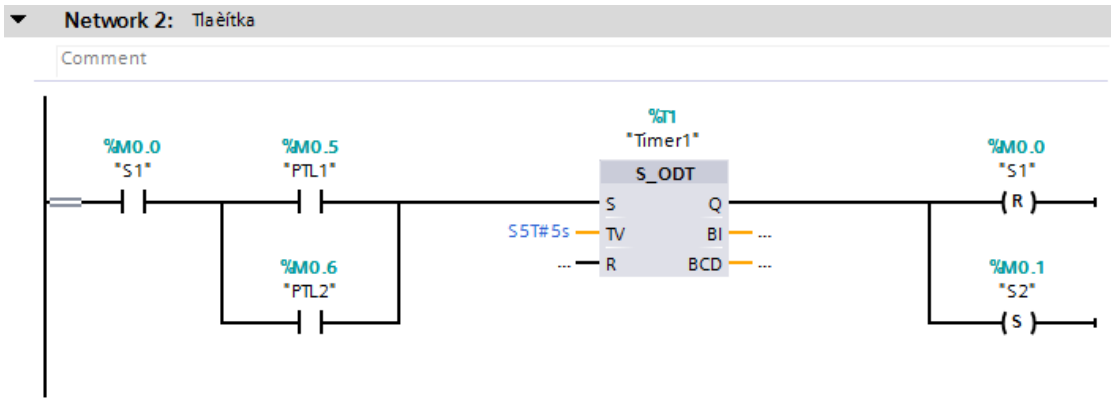
			Address	RLO	Value	Extra
1	A	"S1"	%M0.0	1	1	
2	A(			1		
3	O	"PTL1"	%M0.5	1	1	
4	O	"PTL2"	%M0.6	1	0	
5	)			1		
6	L	S5T#5s	S5T#5s	1	S5T#5S	
7	SD	"Timer1"	%T1	1	S5T#2S_250MS	OV=0
8	NOP	0				
9	NOP	0				
10	NOP	0				
11	A	"Timer1"	%T1	0	0	S5T#2S_25...
12	R	"S1"	%M0.0	0	1	
13	S	"S2"	%M0.1	0	0	

Obrázek 4.12: *Monitoring části programu*

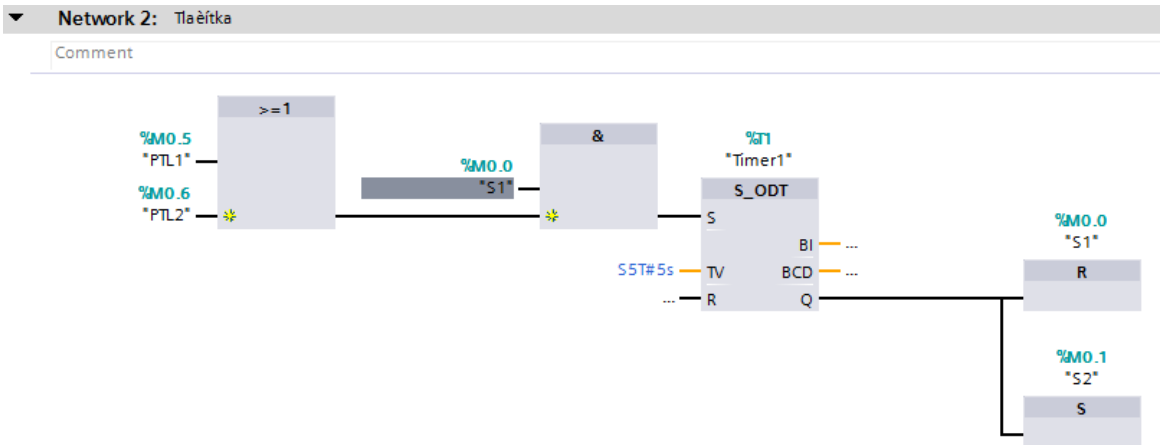
Pokud bude takto vytvořen program pro PLC řady SIMATIC S7-300 a SIMATIC S7-400, lze změnit programovací jazyk z textového STL, na jeden z grafických programovacích jazyků, buďto LAD nebo FBD, změna jazyka pro PLC řady SIMATIC S7-1500 není možná. Změna se provede kliknutím pravým tlačítkem myši na programový blok a zvolením Switch programming language.



Obrázek 4.13: *Změna programovacího jazyka*



Obrázek 4.14: *Změna programovacího jazyka z STL na LAD*



Obrázek 4.15: *Změna programovacího jazyka z STL na FBD*

Závěr

Cíle této bakalářské práce, které spočívaly v nalezení vhodného způsobu popisu sekvenčních úloh a vytvoření nástroje pro zjednodušenou tvorbu sekvenční úlohy byly splněny. Jako nejvhodnější způsob popisu sekvenční úlohy se mi jeví jazyk SFC, který je zanesen v normě IEC 61131-3 a který byl vytvořen speciálně pro sekvenční úlohy, avšak i ostatní metody, které jsou v bakalářské práci zmíněny, jsou vhodné pro popis sekvenčních úloh. Při programování sekvenčních úloh jsem si ověřil, že pokud postupujeme systematicky a úlohu před samotným programováním nejdříve popíšeme podle jednoho z uvedených systematických postupů, tak ve výsledku je program napsán rychleji a především tím minimalizujeme možnost výskytu chyb.

Nástroj pro zjednodušenou tvorbu sekvenční úlohy, který jsem vytvořil v programu Excel 2016, umožňuje programátorovi rychle a účinně vytvořit jednoduché sekvenční úlohy v programu Excel. Nástroj je sice vytvořen jen pro jednoduché sekvenční úlohy, kde musí po každém stavu následovat přechod a stavy se nemohou větvit, avšak je to podle mého názoru způsob, jak práci programátora zjednodušit. Do budoucna by mohl být vytvořen soubor v aplikaci Excel, který by umožňoval řešit jakoukoliv sekvenční úlohu a také by bylo možné vytvářet nejen hlavní programový blok main, ale jakýkoliv programový blok a to včetně funkční bloků a funkcí. Tento nástroj by mohl vycházet ze souboru, který jsem vytvořil, a logika by mohla být podobná.

Při testování jsem zjistil, že takto dokáži program vytvořit velmi rychle a navíc pro samotné vytvoření programu není nutné mít v počítači nainstalovaný nástroj pro programování programovatelných automatů, který je zapotřebí až pro samotné nahrání do PLC, či do simulačního softwaru.

Dále jsem v mé práci ukázal, jak je možné importoval a exportovat části aplikace z, respektive do prostředí TIA Portal, což může také práci programátora velmi zjednodušit, pokud používá stejnou nebo podobnou část programu ve více projektech.

Použitá literatura

[1] SIMATIC STEP 7 Basic (TIA Portal): The Engineering Software for the Basic Controller SIMATIC S7-1200 [online]. [cit. 2017-01-02]. Dostupné z: http://w3.siemens.com/mcms/automation-software/en/tia-portal-software/step7-tia-portal/step7-basic/Pages/Default.aspx

[2] SIMATIC STEP 7 Professional (TIA Portal): The most comprehensive engineering software for demanding controller tasks [online]. [cit. 2017-01-02]. Dostupné z: http://w3.siemens.com/mcms/automation-software/en/tia-portal-software/step7-tia-portal/step7-professional/Pages/default.aspx

[3] TIA Portal Openness Generování projektu. In: TIA Portal Openness Generování projektu [online]. Praha: Siemens, 2016, s. 2-12 [cit. 2017-01-02]. Dostupné z: https://w5.siemens.com/web/cz/cz/corporate/portal/home/produkty_a_sluzby/IADT/tia_na_dosah/Documents/2016_zari/TIAPortalOpenness.pdf

[4] TIA Portal Openness: Introduction and Demo Application: TIA Portal V13 SP1 [online]. Siemens, 2015 [cit. 2017-01-02]. 108716692. Dostupné z: Siemensu

[5] ŠMEJKAL, Ladislav. PLC a automatizace 2: Sekvenční logické systémy a základy fuzzy logiky. Praha: BEN, 2005. ISBN 80-7300-087-3.

[6] TŮMA, Jiří, Renata WAGNEROVÁ, Radim FARANA a Lenka LANDRYOVÁ. Základy automatizace [online]. Ostrava: Ediční středisko VŠB – TUO, 2007 [cit. 2017-01-02]. ISBN 978-80-248-1523-7. Dostupné z: http://www.elearn.vsb.cz/archivcd/FS/Zaut/Skripta_text.pdf

[7] Sekvenční obvody. In: *KATEDRA ELEKTRONIKY* [online]. [cit. 2017-04-20]. Dostupné z: http://fei1.vsb.cz/kat430/data/cmtb/3%20-%20Sekvencni%20obvody.pdf

[8] SIEMENS AG. *SIMATIC STEP 7 Basic V13 SP1: System Manual*. 12/2014. NÜRNBERG, 2015. Dostupné také z: http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s71200/manualy/gsg_step7-basic-v10-5_2014-12_en.pdf

[9] JOHN, Karl-Heinz a Michael TIEGELKAMP. IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids. 2. Berlin: Springer, 2010. ISBN 978-3-642-12014-5.

[10] DORDA, Michal. Úvod do Petriho sítí [online]. [cit. 2017-01-02]. Dostupné z: http://homel.vsb.cz/~dor028/Nekonvencni_metody_1.pdf

[11] KOZIOREK, Jiří, Antonín KUČERA, Jiří HAŠKA a Jan ŠMÍD. Programovatelné automaty a vizualizace řídicích systémů: učební text a návody do cvičení - pracovní verze [online]. Ostrava, 2012 [cit. 2017-01-02].

[12] AUTOMA: časopis pro automatizační techniku. FCC Public, 2012, 18(11). ISSN 1210-9592.

[13] AUTOMA: časopis pro automatizační techniku. FCC Public, 2013, 19(2). ISSN 1210-9592.

[14] AUTOMA: časopis pro automatizační techniku. FCC Public, 2013, 19(6). ISSN 1210-9592.

[15] AUTOMA: časopis pro automatizační techniku. FCC Public, 2011, 17(12). ISSN 1210-9592.

[16] AUTOMA: časopis pro automatizační techniku. FCC Public, 2013, 19(5). ISSN 1210-9592.

[17] AUTOMA: časopis pro automatizační techniku. FCC Public, 2014, 20(3). ISSN 1210-9592.

Seznam příloh

Příloha A: *Kód makra pro spojení sloupců pod sebe*

```
Sub Spojeni_sloupcu()
Dim i As Long
Dim a As Long
Dim b As Long
Dim Sloupce As Long
Dim Radky As Long
Dim PoleA()
Dim PoleB()

 'Smazání hodnot ve sloupcích A, B na listu PomList
 Worksheets("PomList").Range("A:A", "B:B").Clear
 'Překreslování zakázáno
 Application.ScreenUpdating = False
 'Do PoleA jsou uloženy hodnoty
 PoleA = Worksheets("Zadání").Range("A1").CurrentRegion.Value
 'Počet řádků a sloupců v PoleA
 Radky = UBound(PoleA, 1)
 Sloupce = UBound(PoleA, 2)

 'Cyklus for
 For b = 1 To Sloupce Step 2
  For a = 1 To Radky Step 1
   'Podmínky If a změna velikosti PoleB
   If PoleA(a, b) <> "" Then
    If i = 0 Then i = 1: ReDim PoleB(1 To 2, 1 To 1)
    Else i = i + 1: ReDim Preserve PoleB(1 To 2, 1 To i)
    'Zápis do PoleB
    PoleB(1, i) = PoleA(a, b)
    PoleB(2, i) = PoleA(a, b + 1)
    End If
    Next a: Next b
```

```
'Zapsání hodnot na Pomlist
Worksheets("PomList").Cells(1, 1).Resize(i, 2) =
WorksheetFunction.Transpose(PoleB)
'Překreslování povoleno
Application.ScreenUpdating = True
'Zavolání makra Vloz
Call Vloz
End Sub
```

Příloha B: *Kód makra pro vložení částí programu z pomocného listu*

```
Sub Vloz()


Dim Pole()
Dim PomPole()
Dim TypPole()
Dim DataPole()
Dim RozsahDat As Range
Dim RadkyTimeru As Long
Dim RadkyKodu As Long
Dim Hled As Long
Dim i As Long
Dim j As Long
Dim m As Long


  'Rozsah dat na listu TimeryNetworky
  Set RozsahDat =
Worksheets("TimeryNetworky").Range("A1").CurrentRegion
  'Počet řádků na listu TimeryNetworky
  RadkyTimeru = RozsahDat.Rows.Count - 1
  'Počet řádků na listu PomList ve sloupci D
  RadkyKodu = Worksheets("PomList").Cells(Rows.Count,
"D").End(xlUp).Row
  'Vyčištění sloupce G
  Worksheets("PomList").Range("G:G").Clear
  'Zákaz překreslovaní
  Application.ScreenUpdating = False
  'Aktivování listu PomList
  Sheets("PomList").Activate
  'Přečtení údajů a změna velikosti buněk
  PomPole = RozsahDat.Offset(1).Resize(RadkyTimeru).Value
  'Přečtení údajů
  TypPole = RozsahDat.Rows(1).Value
```

```
  'Změna velikosti DataPole
  ReDim DataPole(1 To RadkyKodu, 1 To 1)
  'Přečtení údaj
  DataPole = Cells(1, 4).Resize(RadkyKodu).Value
  'Vypnutí přerušení programu při chybě
  On Error Resume Next
  'Cyklus for
  For i = 1 To RadkyKodu
    'Vyhledání přesné shody DataPole v TypPole
    Hled = WorksheetFunction.Match(DataPole(i, 1), TypPole, 0)
    'Podmínka If. Pokud není nalezena přesná shoda.
    If Err = 0 Then
      j = 1
      'Cyklus Do While Loop
      Do While PomPole(j, Hled) <> ""
        'Podmínka If a změna velikosti Pole
        If m = 0 Then m = 1: ReDim Pole(1 To 1, 1 To 1) Else m = m +
1: ReDim Preserve Pole(1 To 1, 1 To m)
        'Zápis do Pole
        Pole(1, m) = PomPole(j, Hled)
        j = j + 1
        'Opuštění smyčky
        If j > RadkyTimeru Then Exit Do
      Loop

    Else
      'Podmínka If a změna velikosti Pole
      If m = 0 Then m = 1: ReDim Pole(1 To 1, 1 To 1) Else m = m +
1: ReDim Preserve Pole(1 To 1, 1 To m)
      'Zápis do Pole
      Pole(1, m) = DataPole(i, 1)
      'Vymazání Err
      Err.Clear
```

```
    End If
  Next i

  'Zapsání hodnot
  Cells(1, 7).Resize(m) = WorksheetFunction.Transpose(Pole)
  'Aktivování listu Zadání
  Sheets("Zadání").Activate
  'Povolení překreslovaní
  Application.ScreenUpdating = True
  'Zavolání makra VlozKonec
  Call VlozKonec
End Sub
```

Příloha C: *Kód makra pro uložení kódu ve formátu pro PLC řady SIMATIC S7-1500*

```
Sub UkladaniKodu()
Dim Novy As Workbook

  'Zvolení cesty kde se uloží soubor
  Cesta = Application.GetSaveAsFilename(, "Soubor s příponou
AWL,*.awl")

  'Zákaz překreslovaní
  Application.ScreenUpdating = False
  'Zákaz upozornění
  Application.DisplayAlerts = False

  'Podmínka pokud je zvolena neplatná cesta
  If Cesta = False Then
  Exit Sub
  Else
  End If
  'Zkopírování listu
  Worksheets("MainKod1500").Copy
  'Aktivování nového pracovního sešitu do kterého se vloži
zkopírovaný list
  Set Novy = ActiveWorkbook
  'Uložení a zavření nového pracovního sešitu
  With Novy
    .SaveAs Filename:=Cesta, FileFormat:=xlTextPrinter
    .Close
  End With
  'Povolení upozornění
  Application.DisplayAlerts = True
  'Povolení překreslování
  Application.ScreenUpdating = True
End Sub
```

Příloha D: *Kód makra pro uložení tabulky proměnných*

```
Sub UkladaniTagu()
Dim Novy As Workbook
Dim NovyList As Worksheet

  'Zvolení cesty kde se uloží soubor
  Cesta = Application.GetSaveAsFilename(, "Soubor s příponou
XLSX,*.xlsx")

  'Zákaz překreslovaní
  Application.ScreenUpdating = False

  'Podmínka pokud je zvolena neplatná cesta
  If Cesta = False Then
  Exit Sub
  Else
  End If

  'Zkopírování listu
  Worksheets("PLC Tags").Copy
  'Aktivování nového pracovního sešitu a listu, na který se vloži
zkopírovaný list
  Set Novy = ActiveWorkbook
  Set NovyList = Novy.Sheets(1)

  'Pojmenování nového listu
  With NovyList
    .UsedRange.Value = .UsedRange.Value
    .Name = "PLC Tags"
  End With

  'Uložení a zavření nového pracovního sešitu
  With Novy
```

```
    .SaveAs Filename:=Cesta, FileFormat:=xlOpenXMLWorkbook
    .Close
  End With

  'Povolení překreslování
  Application.ScreenUpdating = True

End Sub
```

Příloha E: *Příloha na CD*

Struktura přílohy na CD:

- Marek_Jeriga__JER0031_BP.pdf – bakalářská práce ve formátu PDF/A
- BP_MJ_kod.xlsm – vytvořený nástroj s kódem, který posloužil k otestování nástroje
- BP_MJ_prazdne.xlsm – nevyplněný nástroj
- BP_PLC – testovací úloha vytvořena pomocí nástroje
- BP_1500.awl – kód pro PLC řady SIMATIC S7-1500 vygenerovaný pomocí nástroje
- PLCtags.xlsx – proměnné vygenerované pomocí nástroje